Spring 1-1-2012

A Unified Microwave Radiative Transfer Model with Jacobian for General Stratified Media

Miao Tian
University of Colorado at Boulder, miao.tian@colorado.edu

A Unified Microwave Radiative Transfer Model with Jacobian for General Stratified Media

by

Miao Tian

B.S., University of Electronic Science and Technology of China, Chengdu, Sichuan, China, 2003

M.S., University of Tulsa, Oklahoma, U.S.A., 2005

M.S., University of Colorado at Boulder, Colorado, U.S.A., 2008

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical, Computer, and Energy Engineering

2012

This thesis entitled:
A Unified Microwave Radiative Transfer Model with Jacobian for General Stratified Media
written by Miao Tian
has been approved for the Department of Electrical, Computer, and Energy Engineering

Albin J. Gasiewski

Dr. Alexander Voronovich

Prof. Edward Kuester

Prof. Zoya Popovic

Prof. Dejan Filipovic

Prof. Nikolay Zabotin

Prof. Leung Tsang

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Tian, Miao (Ph.D., Passive Microwave Remote Sensing)

A Unified Microwave Radiative Transfer Model with Jacobian for General Stratified Media

Thesis directed by Prof. Albin J. Gasiewski

A unified microwave radiative transfer (UMRT) model is developed for rapid, stable and accurate level-centric calculation of the thermal radiation emitted from any geophysical medium comprised of planar layers of either densely or tenuously distributed, moderately sized spherical scatterers. The formulation includes rapid calculation of the tangent linear relationship (i.e., Jacobian) between the observed brightness temperature and any relevant radiative and geophysical layer parameters, such as the scattering and absorption coefficients, temperature, temperature lapse rate, and medium layer thickness.

UMRT employs a rapid multistream scattering-based discrete ordinate eigenanalysis solution with a layer-adding algorithm stabilized by incorporating symmetrization of the discretized differential radiative transfer equations and analytical diagonalization and factorization of the resulting symmetric and positive definite matrices. It is based on the discrete ordinate tangent linear radiative transfer model of Voronovich et al. (2004), but extended to include both Mie and dense media scattering theories and employ refractive layers.

Other nontrivial extensions are: 1) exact modeling of linearized temperature profiles and resulting radiation streams across medium layers, 2) compensation for refracted radiation streams using Snell's law, the Fresnel reflectivity and transmissivity coefficients, and a cubic spline interpolation matrix, and 3) seamless calculation of associated Jacobians for both sparse and dense medium parameters.

Details of the UMRT Jacobian formulation are presented. The entire formulation has been programmed in Matlab and validated through both energy conservation and numerical Jacobian intercomparisons. Comparisons of the upwelling brightness temperatures over dry snow and ice from simulations and field measurements are presented and discussed.

To my parents

To my wife, Ziqi

To my son, Alexander

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. A.J. Gasiewski for his guidance, patience, support, and encouragement throughout my Ph.D. studies at University of Colorado, Boulder. I am indebted to him for sharing with me his extensive knowledge, experience and rigorous approach to the study of science and engineering. In particular, his spirit of thoroughness in research greatly impacted me.

I'm grateful for Dr. A. Voronovich for his valuable suggestions on developing the unified microwave radiative transfer model of this research, and Dr. V. Zavorotny for the help on my first porject of the L-band radiometer calibration started in 2006. I'm also grateful for Profs. L. Tsang, E. Kuester, Z. Popovic, D. Filipovic, U. Herzfeld, and N. Zabotin for serving on my Dissertation Committee, and for their enthusiastic responses and helpful suggestions concerning my work.

Among my friends at CU, I want to thank Drs. Si Liu, Yuqi Wu and Sandeep Kumar. Si and Yuqi were helpful as my loyal audience and frequently offered valuable suggestions on math problems arose in my work. Kumar's long-standing discussions with me on aspects of radiometer calibration and radiative transfer theory have been invaluable. Other friends to acknowledge that, over the years, contributed positively to my research are: Mr. David Kraft, Mr. Eric McIntyre, Mr. Zixu Zhu, Mr. Jun Xiao, Dr. Qianli Mu, and Mr. Michael Zucker.

Last but foremost, I would like to thank my family, my parents Huaicong Tian and Xiaoduan Li for their unfailing love and confidence put in me over the last thirty years, and in particular, my wife Ziqi Chu, without her love, support and sacrifice, I cannot endure this long journey. I greatly appreciate her solo efforts in her pregnancy during the last nine months while I was completing my dissertation. Also, I'm grateful for the patience and many helpful life suggestions from my parents-in-law, Ming Chu and Yan Zhang.

In addition, financial support from Colorado Engineering, Inc. on project W31P4Q-11-C-0054 for a "Terahertz Atmospheric and Ionospheric Propagation, Absorption and Scattering (TAIPAS) Model" sponsored by the U.S. Army Research Office, is acknowledged.

Contents

Chapter

List of Tables

Table

List of Figures

Figure

Chapter 1

Introduction

1.1 Motivations

Relative to visible and infrared sensing systems, passive microwave sensors have been proven valuable for a wide range of environmental remote sensing applications due to their ability to observe through clouds and into surfaces with a predictable and wide range of probing depths, along with the ability to continuously monitor environmental variables throughout the diurnal cycle and under most weather conditions. Brightness temperatures measured using passive microwave sensors are useful for retrieval of geophysical variables such as ocean surface salinity [1], ocean surface wind direction and velocity [2, 3, 4], stratiform precipitation [5, 6, 7], ice path and thickness [8, 9, 10], water vapor [11, 12], soil moisture [13, 14], and others. Thus, the development of a general, well validated microwave thermal emission model applicable to all of these problems is significant for both research and operational purposes.

Moreover, obtaining information on the tangent linear relationship (i.e., Jacobian) between the observed brightness temperatures and relevant radiative and geophysical parameters, such as scattering (κ_s) and absorption (κ_a) coefficients, medium temperature (T_o), temperature lapse rate (γ_t), medium layer thickness (d), and others, under arbitrary scattering and absorbing conditions is important to enhance the usage efficiency of measured passive microwave data via data assimilation as well as better understand the fundamental emission sensitivity relations and thus is very valuable for nonlinear retrievals wherein the background atmospheric and surface states are determined from a numerical weather prediction model, climatology, or other related means. In addition, the Jacobians

also prove useful in detecting subtle or more obvious modeling errors by intercomparing to radiative transfer models.

Currently, a major challenge in passive microwave remote sensing is the accurate and fast forward numerical modeling of the bulk electromagnetic scattering properties and thermal emission of any geophysical medium consisting of layers of soil, water, seawater, snow, ice, rain, cloud, fog, etc., along with accurate and rapid calculation of the Jacobian matrix. Of importance in any such model is accuracy, numerical stability, computational speed, applicability to both dense and tenuous scattering media, and the capability to produce the Jacobian matrix for radiance assimilation purposes.

In radiative transfer theory, for cases where scattering exists there are three primary solution techniques to solve the differential radiative transfer equation (DRTE) for the four Stokes parameters: 1) the iterative method [15, 16], 2) the discrete ordinate eigenanalysis (DOE) method [17, 18, 19, 20], and 3) the Monte Carlo method [21]. Among these, the iterative method is applicable to low albedo cases or thin layers, and the Monte Carlo method lacks physical insight and convergence criteria. The DOE method with layer-adding algorithm is widely used due to its applicability to layers of arbitrary albedo. In the DOE method, the continuum of propagation directions is described by a finite number of quadrature angles. The resulting system of equations is solved by eigenanalysis, and medium inhomogeneity is accommodated by layer-adding.

The basic DOE solution for a multilayer structure under the planar stratified approximation follows the formulation developed by Stamnes and Swanson, 1981 [18]. In this work a matrix-operator method to solve the DRTE as an eigenvalue problem and technique to reduce the order of the problem by a factor of two were devised. In 1986, Nakajima and Tanaka [19] introduced the decomposition of a symmetric transition matrix to provide a nearly-stable numerical solution for the DRTE. Matrix operator representations of the reflection and transmission matrices in the multilayer stack were also introduced in their algorithm. In 1988, the DOE model was summarized by Stamnes [20] for general use in planar multilayer multiple scattering media.

Although the above models have been successful over the years, there remained two ma-

jor problems within the DOE formulation: 1) analytic functions of matrices were required to be computed using Taylor series expansions. For example, for a sufficiently small transition matrix argument $\overline{\overline{A}}\,\overline{\overline{B}}h$, one can calculate the cosine hyperbolic operator of this argument as:

$$\cosh\left(\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}h\right) = 1 + \frac{\overline{\overline{A}}\,\overline{\overline{B}}}{2!}h^2 + \frac{\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^2}{4!}h^4 + \cdots \tag{1.1}$$

The above expansion generally requires too many terms for practical implementation. Accordingly, the accuracy of the DOE solution is compromised by accumulated roundoff errors. 2) A second issue is the well-known matrix inversion instability associated with implementation of the DOE method for high albedo, high opacity and thick layers. These two attributes have historically limited the applicability of the DOE method.

To circumvent these problems Voronovich **et al.**, (2004) [22] developed the discrete ordinate tangent linear radiative transfer model (DOTLRT) based on symmetrization of the DRTE and analytical diagonalization and factorization of the resulting symmetric and positive definite matrices to provide inherent computational stability and high computational efficiency for all matrix operations required by the DOE method. The core DOTLRT procedure requires that both transition matrices $\overline{\overline{A}}$ and $\overline{\overline{B}}$ are symmetric and positive definite, in which case any arbitrary analytic function g operated on the matrix product $\overline{\overline{A}}\,\overline{\overline{B}}$ can be readily calculated. Specifically, applying symmetry the matrix $\overline{\overline{A}}$ can be represented as

$$\overline{\overline{A}} = \overline{\overline{M}}_1 \overline{\overline{\Lambda}}_1 \overline{\overline{M}}_1^T \tag{1.2}$$

where $\overline{\overline{M}}_1$ is an orthogonal matrix consisting of eigenvectors of $\overline{\overline{A}}$ having the following characteristics:

$$\overline{\overline{M}}_1 \overline{\overline{M}}_1^T = \overline{\overline{M}}_1 \overline{\overline{M}}_1^{-1} = \overline{\overline{I}} \tag{1.3}$$

where $(\cdot)^T$ denotes the matrix transpose and $\overline{\overline{I}}$ is the identity matrix. In (1.3), $\overline{\overline{\Lambda}}_1$ is a diagonal matrix of associated eigenvalues. Since $\overline{\overline{A}}$ is positive definite, the eigenvalues are positive ($\left\{\overline{\overline{\Lambda}}_1\right\}_{ii} > 0$),

which guarantees that values of $\overline{\overline{\Lambda}}_1^{\pm\frac{1}{2}}$ are all positive real. Similarly, another set of eigenvalue and eigenvector matrices can be defined using the matrix $\overline{\overline{B}}$ and $(\overline{\overline{\Lambda}}_1, \overline{\overline{M}}_1)$ in (1.2) as

$$\overline{\overline{\Lambda}}_1^{\frac{1}{2}}\overline{\overline{M}}_1^T\overline{\overline{B}}\,\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{\frac{1}{2}} = \overline{\overline{M}}_2\overline{\overline{\Lambda}}_2\overline{\overline{M}}_2^T \tag{1.4}$$

Using (1.2) and (1.4), the product of $\overline{\overline{A}}\,\overline{\overline{B}}$ can be calculated as

$$\overline{\overline{A}}\,\overline{\overline{B}} = \left(\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{\frac{1}{2}}\overline{\overline{M}}_2\right)\overline{\overline{\Lambda}}_2\left(\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\overline{\overline{M}}_2\right)^T \tag{1.5}$$

As a result,

$$g\left(\overline{\overline{A}}\,\overline{\overline{B}}\right) = \left(\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{\frac{1}{2}}\overline{\overline{M}}_2\right)g\left(\overline{\overline{\Lambda}}_2\right)\left(\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\overline{\overline{M}}_2\right)^T \tag{1.6}$$

for any analytical matrix function g. Moreover, by incorporating the derivative chain rule using first-order perturbations of the eigenvalues and eigenvectors of a symmetric matrix, DOTLRT provides rapid numerical calculation of the associated Jacobians between the observed brightness temperature and all relevant radiative and geophysical parameters.

Although the DOTLRT algorithm provides a stable, fast and accurate solution to the DRTE, it was originally developed for atmospheric simulation, in which scattering hydrometeors are sparse (e.g., rain, fog, cloud, and aerosols). It also is based on a single polarization using the Henyey-Greenstein (HG) phase matrix approximation for a planar multilayer structure with non-refracting layers. Finally, it is based on layers with constant physical temperature. These attributes have limited its application, especially for cases of dense media (e.g., snow, ice, soil, etc.,) and thick atmospheric or surface layers with strong temperature gradients.

1.2 Contributions

In this thesis, a new "unified microwave radiative transfer" (UMRT) model is formulated to extend DOTLRT in all of the above areas. This model can be applied to widely varying types of media for both forward radiative transfer and radiance assimilation purposes. Within UMRT media

layers are seamlessly partitioned into two categories, which are treated distinctively as follows: 1) sparse medium layers, in which scatters are loosely distributed and independent scattering is dominant, and 2) dense medium layers, in which scatters occupy significant volume fraction and volumetric scattering is dominant.

For sparse medium layers, the cross-polarization between the vertical and horizontal radiation intensities is considered by using the reduced Mie phase matrix. A proof of the symmetry and positive definite nature of the Mie phase matrix is developed to ensure the applicability of the stable matrix operation formulation of DOTLRT. Assuming independent scattering, UMRT sparse medium layers are parametrized by sets of particle size distribution functions for each of the different scatterer phases, for example, liquid spheres [23, 24, 25, 26, 27, 28], snowflakes [29, 30], ice spheres [31, 32, 33], etc. Calculations of the associated extinction, scattering and absorption coefficients, and phase matrices are performed for each of these phases.

For dense medium layers, the dense media radiative transfer theory (DMRT) is applied within UMRT. The DMRT theory with the quasi-crystalline approximation (QCA) was developed by Tsang and his colleagues beginning in the early 1980s [34, 35, 17, 21, 36]. In UMRT, a recent (2007) version of the DMRT-QCA model by Tsang **et al**. [37] is used. This model uses a sticky particle assumption for moderately sized (i.e., Mie-scale) spherical particles. In this model, the adhesion and aggregation of the sticky particles are simulated by using sticky pair distribution functions based on the Percus-Yevick approximation. As used within UMRT the reduced DMRT-QCA phase matrix is included and its symmetry properties are identified. The associated absorption and scattering coefficients are calculated under the DMRT framework.

Other nontrivial extensions in UMRT are:

(1) UMRT provides an exact solution for a piecewise linear temperature profile, thus extending the accuracy of the previous single-layer DRTE solution.

(2) UMRT employs a planar multilayer structure with refracting layers that properly accommodate both reflection and refraction at layer interfaces by applying Fresnel's and Snell's

laws to all radiation streams. The effects of background refractivity variations are thus compensated by excluding transmission beyond critical angles and applying a cubic spline interpolation to the remaining refracted/transmitted streams.

(3) UMRT includes a revised Jacobian procedure to accommodate its refractive dense multilayer framework.

During the development of UMRT, we realized that for a single particle using a spherical or non-spherical assumption will indeed yield significant brightness differences, and also realized that for certain non-spherical cases such as cylinders and spheroids, the associated scattering problems have been addressed [17]. The reason we do not include the non-spherical cases in this work is that an intrinsic problem with non-spherical particle theory is that it will introduce more parameters, e.g., aspect ratios, orientation distributions, etc., into the problem. However, these parameters for non-spherical particles are difficult to actually measure, and if particles are randomly oriented in a medium the non-spherical and spherical cases usually give similar results in the polydispersed scattering case. We thus wanted to focus this modeling effort on the most basic particle type and study the impact of other issues, such as polarization, refraction, discretization, integration accuracy, computation speed and fast Jacobian development.

Our discussion of rapid computation capability directly follows that for DOTLRT in the Section IX of [22]. In UMRT, the number of operations required for calculation of both the brightness temperature profile and associated Jacobian for all stream angles (M angles) is NM^3, where N is the total number of layers. Since the same complexity applies we do not bother to belabor the discussion again. As the previous argument goes: 1) for a conventional DOE solution with a divided difference Jacobian, the number of operations required is N^2, and 2) for an iterative perturbation solution the number of operations is N^3. Normally, $N \gg M$, therefore DOTLRT and UMRT are rapid models in this regard.

1.3 Outlines

A block diagram of the entire formulation of the unified microwave radiative transfer model (UMRT) is illustrated in Fig. 1.1.

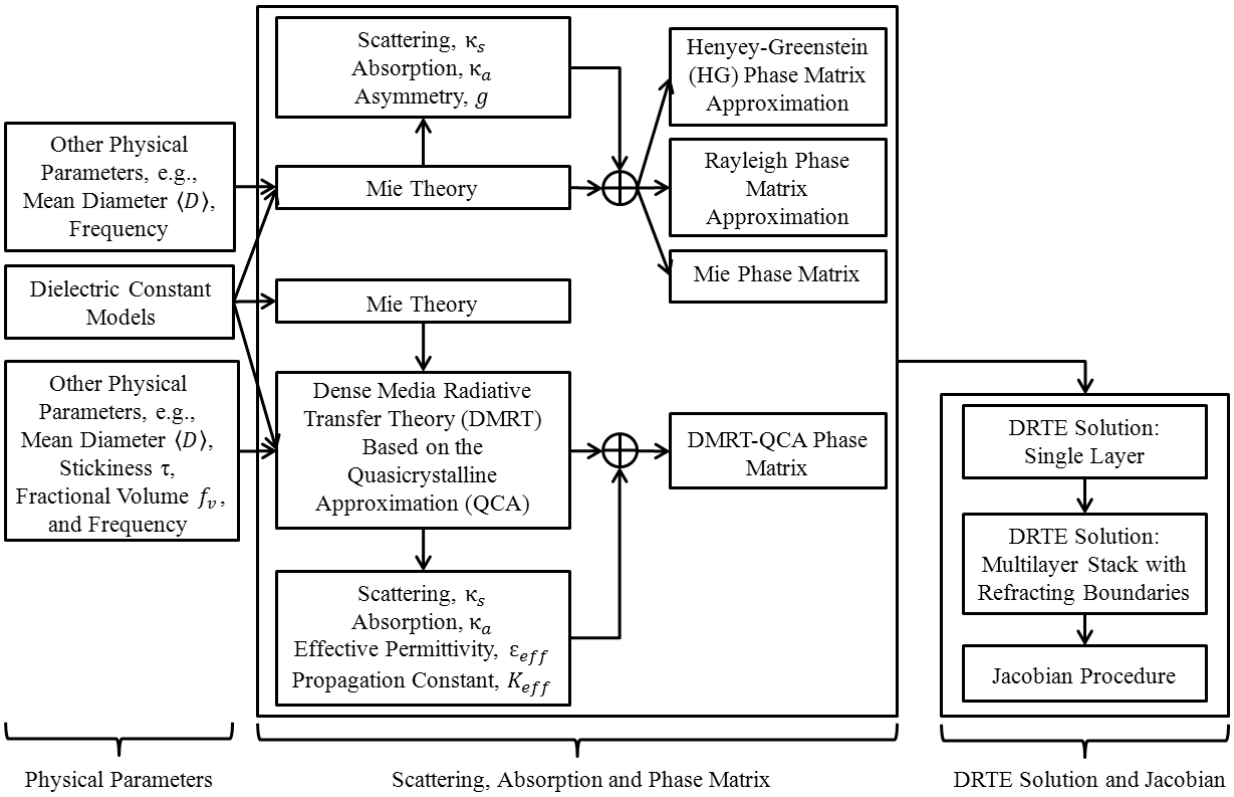


Figure 1.1: Block diagram of the entire UMRT Jacobian formulation.

It is seen that a major step within the UMRT formulation is the calculation of various types of phase matrix including the Henyey-Greenstein (HG), Rayleigh, Mie, and the DMRT-QCA, along with using a set of environmental and medium (physical and electric) parameters including physical temperature T, frequency f, particle mean diameter $\langle D \rangle$, medium permittivity ε, volume fraction of scatterers f_v, and stickiness parameter τ as input. After this step, UMRT will first provide solutions of emission vector, reflection and transmission matrices, and relevant Jacobian matrices for each and every single layer within a N-layer stack, and then provide solutions for the multilayer stack at each and every level in the form of explicit recurrence relations.

The thesis is organized as follows: Chapter 2 contains a brief introduction of the microwave radiative transfer theory along with a summary of the equations used in UMRT for calculations of the four types of phase matrix and (correspondingly) their associated extinction, scattering and absorption coefficients. Numerical results and intercomparison of these phase matrices and coefficients are provided and discussed in this chapter. Chapter 3 contains the slab formulation of UMRT for a general planar stratified structure without refractive boundaries. Several numerical examples of the UMRT single layer model are provided and discussed at the end of this chapter. Chapter 4 contains the extended UMRT Jacobian formulation based on the multilayer structure with refractive boundaries prescribed in beginning of this chapter. Chapter 5 contains numerical validation of the entire UMRT Jacobian formulation and comparisons of the upwelling brightness temperatures over dry snow and ice from simulations and field measurements. Chapter 6 contains a short summary and future work direction.

Chapter 2

Microwave Radiative Transfer Theory

Theoretical physics of microwave radiative transfer in a general planar-stratified medium comprised of layers of either densely or tenuously distributed, moderately sized spherical particles are reviewed, along with the equations for computing four types of phase matrix (i.e., the Henyey-Greenstein, Rayleigh, Mie, and DMRT) and (correspondingly) their associated extinction, scattering and absorption coefficients. Symmetry nature of the phase matrix is discussed. Numerical results of the four phase matrices and their associated coefficients based on the azimuthal symmetry of the planar-stratified approximation are presented and discussed.

2.1 Vector Radiative Transfer Theory

2.1.1 Differential Radiative Transfer Equation

Radiative transfer theory has been addressed in a number of books: Chandrasekhar, 1960 [38]; Ishimaru, 1978 [39]; Tsang **et al.**, 1985 [40]; Ulaby **et al.**, 1990 [41]; Janssen **et al.**, 1993 [15]; Fung, 1994 [42]; Tsang **et al.**, 2000 [17, 21, 36]; Matzler **et al.**, 2006 [43]. In general, it is an important method to treat the propagation of microwave electromagnetic fields within a medium affected by both scattering and absorption due to the presence of randomly distributed particles. Moreover, the absorbing particles also cause emission that equals absorption under the assumption of local thermodynamic equilibrium between the particles and the surrounding environment. The above characteristics of scattering, absorption and emission are governed by an integro-differential equation, called the differential radiative transfer equation (DRTE). In this section, the DRTE for

a general medium is introduced, along with the three DRTE constituents: phase matrix, extinction matrix and absorption vector.

Generally, an arbitrarily polarized electromagnetic radiation field can be described by a specific intensity, which is a four-element modified Stokes' vector:

$$\overline{I}\left(\overline{r},\hat{s},f\right)=\begin{bmatrix} I_v \\ I_h \\ U \\ V \end{bmatrix} \triangleq \frac{1}{\eta}\begin{bmatrix} \left\langle |E_v|^2 \right\rangle \\ \left\langle |E_h|^2 \right\rangle \\ 2\mathrm{Re}\left\langle E_v E_h^* \right\rangle \\ 2\mathrm{Im}\left\langle E_v E_h^* \right\rangle \end{bmatrix} \quad \mathrm{W/_{m^2-sr-Hz}} \tag{2.1}$$

where $\overline{I}\left(\overline{r},\hat{s},f\right)$ is the specific intensity evaluated at position $\overline{r}$, frequency f and along certain propagation direction denoted by a unit vector $\hat{s}$, I_v and I_h are the spectral intensities of the vertically and horizontally polarized electromagnetic field components E_v and E_h, (respectively), U and V are the in-phase and quadrature covariances between the vertical and horizontal field components (respectively), and η is the wave impedance. In Eq. (2.1), $\mathrm{Re}\left\langle\cdot\right\rangle$ means real part of $\left\langle\cdot\right\rangle$ and $\mathrm{Im}\left\langle\cdot\right\rangle$ means imaginary part of $\left\langle\cdot\right\rangle$.

Using the property of incoherent addition of Stokes' parameters, the DRTE that governs the propagation of specific intensity, has the following vector form [15]:

$$\begin{aligned} \hat{s}\cdot\nabla\overline{I}\left(\overline{r},\hat{s},f\right)= & -\overline{\overline{\kappa}}_e\left(\overline{r},\hat{s},f\right)\cdot\overline{I}\left(\overline{r},\hat{s},f\right)+\overline{\kappa}_a\left(\overline{r},-\hat{s},f\right)\cdot B\left(T\left(\overline{r}\right),f\right) \\ & +\int\overline{\overline{P}}\left(\overline{r},\hat{s},\hat{s}',f\right)\cdot\overline{I}\left(\overline{r},\hat{s}',f\right)d\Omega' \end{aligned} \tag{2.2}$$

where $\overline{\overline{\kappa}}_e$ is the extinction matrix that describes the attenuation of specific intensity due to absorption and scattering, $\overline{\kappa}_a$ is the absorption vector, $B\left(T\left(\overline{r}\right),f\right)$ is the spectral intensity function represents the energy emitted by a blackbody at thermodynamic temperature $T\left(\overline{r}\right)$ and frequency **f**, the product of $\overline{\kappa}_a\cdot B\left(T\left(\overline{r}\right),f\right)$ is the emission vector due to local thermodynamic equilibrium, and $\overline{\overline{P}}\left(\overline{r},\hat{s},\hat{s}',f\right)$ is the phase matrix that describes the coupling of intensities from all directions and polarizations due to scattering [17, 15]. Since the specific intensity $\overline{I}\left(\overline{r},\hat{s},f\right)$ is a four-element modified Stokes' vector, the phase matrix and extinction matrix are both 4×4 matrices, and the

absorption vector is a 4×1 column matrix. For non-spherical particles the extinction matrix is nondiagonal and the four elements of the absorption vector are all nonzero.

The specific intensity and thermodynamic temperature is connected by the Planck radiation law:

$$I = \mu\epsilon \frac{hf^3}{e^{hf/KT} - 1} \tag{2.3}$$

where μ and ϵ are permittivity and permeability of a medium (respectively), h is Planck's constant (6.634×10^{-34} joule-sec), K is Boltzmann's constant (1.38×10^{-23} joule/Kelvin), and T is a thermodynamic temperature (in Kelvin). Applying Rayleigh-Jean's approximation (valid for frequencies up to ~ 300 GHz and terrestrial temperatures), in free space Eq. (2.3) is simplified as

$$I \approx \frac{KT}{\lambda^2} \tag{2.4}$$

where λ is the free-space wavelength. The Rayleigh-Jean's approximation is sufficient for most microwave applications, though doing so does not restrict application of the full Planck's function. Applying Eq. (2.4) to Eq. (2.1), the specific intensity $\overline{I}(\overline{r}, \hat{s}, f)$ can be scaled to an equivalent brightness temperature vector $\overline{T}_B$:

$$\overline{T}_B \equiv \frac{\lambda^2}{K}\overline{I} = \begin{bmatrix} T_{Bv} \\ T_{Bh} \\ T_U \\ T_V \end{bmatrix} \quad \text{in Kelvin} \tag{2.5}$$

2.1.2 Phase Matrix, Extinction Matrix and Absorption Vector

2.1.2.1 Scattering Function Matrix and Stokes Matrix

Assuming independent scattering, expressions for the phase matrix, extinction matrix and absorption vector can all be expressed in terms of the scattering function matrix $\overline{\overline{F}}(\Theta)$, which

describes the amplitude and polarization of an incident plane wave scattered by a single particle to any direction at a distance r:

$$\overline{E}_s = \frac{e^{-jkr}}{r}\overline{\overline{F}}(\Theta) \cdot \overline{E}_i \tag{2.6}$$

where $k = \frac{2\pi}{\lambda}$ is the wavenumber in air, and $\overline{E}_i$ and $\overline{E}_s$ are the complex electric fields of the incident and scattered waves, respectively. A particle-based coordinate system (Fig. A.1, taken from Fig. 1.1.3 in [17]) is assumed in Eq. (A.1) and it is defined based on the scattering plane consisting of $\hat{k}_i$ and $\hat{k}_s$, which are the incident and scattered directions, respectively. In the scattering plane, the angle between $\hat{k}_i$ and $\hat{k}_s$ is Θ, called the forward scattering angle. In the particle-based coordinate system, the scattering by the particle is described by two orthonormal unit systems $\left(\hat{1}_i, \hat{2}_i, \hat{k}_i\right)$ and $\left(\hat{1}_s, \hat{2}_s, \hat{k}_s\right)$. Relations between the two unit vector sets can be found in [17] and also included in Appendix A.

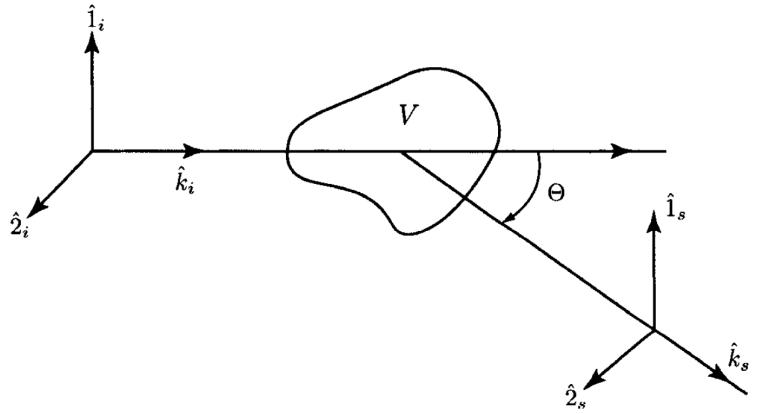


Figure 2.1: Geometry of the particle-based coordinate system. The scattering plane contains $\hat{k}_i$ and $\hat{k}_s$ and the angle between $\hat{k}_i$ and $\hat{k}_s$ is Θ.

Expressing the plane waves $\overline{E}_i$ and $\overline{E}_s$ in the vertical-horizontal polarization basis, Eq. (A.1) is written as

$$\begin{bmatrix} E_{vs}(\Theta) \\ E_{hs}(\Theta) \end{bmatrix} = \frac{e^{-jkr}}{r} \begin{bmatrix} f_{vv}(\Theta) & f_{vh}(\Theta) \\ f_{hv}(\Theta) & f_{hh}(\Theta) \end{bmatrix} \begin{bmatrix} E_{vi}(\Theta) \\ E_{hi}(\Theta) \end{bmatrix} \tag{2.7}$$

where $a, b = v, h$. Using the definitions in Eq. (2.1), the scattered modified Stokes' parameters can be related to the incident modified Stokes' parameters from Eq. (A.2):

$$\overline{I}_s(\Theta) = \frac{1}{r^2}\overline{\overline{L}}(\Theta) \cdot \overline{I}_i(\Theta) \tag{2.8}$$

where $\overline{I}_s(\Theta)$ and $\overline{I}_i(\Theta)$ are column vectors

$$\overline{I}_s(\Theta) = \begin{bmatrix} I_{vs}(\Theta) \\ I_{hs}(\Theta) \\ U_s(\Theta) \\ V_s(\Theta) \end{bmatrix} \tag{2.9}$$

$$\overline{I}_i(\Theta) = \begin{bmatrix} I_{vi}(\Theta) \\ I_{hi}(\Theta) \\ U_i(\Theta) \\ V_i(\Theta) \end{bmatrix} \tag{2.10}$$

and $\overline{\overline{L}}(\Theta)$ is the Stokes matrix [17].

$$\overline{\overline{L}}(\Theta) = \begin{bmatrix} |f_{vv}|^2 & |f_{vh}|^2 & \mathrm{Re}\{f_{vh}^* f_{vv}\} & -\mathrm{Im}\{f_{vh}^* f_{vv}\} \\ |f_{hv}|^2 & |f_{hh}|^2 & \mathrm{Re}\{f_{hh}^* f_{hv}\} & -\mathrm{Im}\{f_{hh}^* f_{hv}\} \\ 2\mathrm{Re}\{f_{vv} f_{hv}^*\} & 2\mathrm{Re}\{f_{vh} f_{hh}^*\} & \mathrm{Re}\{f_{vv} f_{hh}^* + f_{vh} f_{hv}^*\} & -\mathrm{Im}\{f_{vv} f_{hh}^* - f_{vh} f_{hv}^*\} \\ 2\mathrm{Im}\{f_{vv} f_{hv}^*\} & 2\mathrm{Im}\{f_{vh} f_{hh}^*\} & \mathrm{Im}\{f_{vv} f_{hh}^* + f_{vh} f_{hv}^*\} & \mathrm{Re}\{f_{vv} f_{hh}^* - f_{vh} f_{hv}^*\} \end{bmatrix} \tag{2.11}$$

2.1.2.2 General Forms of Phase Matrix, Extinction Matrix and Absorption vector

Under the independent scattering assumption, the phase matrix is subsequently computed by averaging the Stokes matrix over an ensemble of particles of varying geometry in terms of varying size, shape, orientation, and dielectric constitution:

$$\overline{\overline{P}}(\Theta) = n_o \left\langle \overline{\overline{L}}(\Theta) \right\rangle \tag{2.12}$$

where n_o is the total particle density and $\langle\cdot\rangle$ denoted ensemble average. From [17, 15], if distribution of the particles was known, then phase matrix can be calculated by integrating the Stokes matrix with respect to the specified particle size distribution function, $n(D)$.

$$\overline{\overline{P}}(\Theta) = \int_0^\infty \left\langle \overline{\overline{L}}(\Theta) \right\rangle \cdot n(D)\, dD \tag{2.13}$$

where D is the sphere diameter. In this research, a few well-known particle size distribution functions for various hydrometers: raindrops, snowflakes and ice particles, are studied and programmed (can be found in the Matlab programs listed at the end of this thesis) for the background research purposes. Specifically, in this thesis two representative particle size distributions are used to produce numerical examples: 1) the Marshall-Palmer (MP) size distribution function [23] for liquid water raindrops:

$$n(D) = N_o e^{-\Lambda D} \text{ where } \begin{cases} N_o = 8 \times 10^3 & \left(\mathrm{m}^{-3} \cdot \mathrm{mm}^{-1}\right) \\ \Lambda = 4.1 R^{-0.21} & \left(\mathrm{mm}^{-1}\right) \end{cases} \tag{2.14}$$

where R is the rain rate in mm/hr, and 2) the Sekhon-Srivastava (SS) size distribution function [29] for snowflakes:

$$n(D) = N_o e^{-\Lambda D} \text{ where } \begin{cases} N_o = 2.50 \times 10^3 R^{-0.94} & \left(\mathrm{m}^{-3} \cdot \mathrm{mm}^{-1}\right) \\ \Lambda = 2.29 R^{-0.45} & \left(\mathrm{mm}^{-1}\right) \end{cases} \tag{2.15}$$

where R is the equivalent liquid-water precipitation rate in mm/hr. More details of relevant $n(D)$ functions for atmospheric hydrometeors can be found in [15, 44] and many other related publications.

It needs to point out that since multiple volumetric scattering is dominant in dense medium, according to the DMRT-QCA theory, the particle size distribution function used in Eq. (2.13) should be replaced by a structure factor that is estimated from the Percus-Yevick pair distribution function. Details of the four aforementioned phase matrices are given in §2.3.

For non-spherical particles, the extinction matrix is formulated in terms of the forward scattering amplitudes as [17]:

$$\overline{\overline{\kappa}}_e = \frac{2\pi n_o}{k}\begin{bmatrix} 2\mathrm{Im}\{f_{vv}\} & 0 & \mathrm{Im}\{f_{vh}\} & -\mathrm{Re}\{f_{vh}\} \\ 0 & 2\mathrm{Im}\{f_{hh}\} & \mathrm{Im}\{f_{hv}\} & \mathrm{Re}\{f_{hv}\} \\ 2\mathrm{Im}\{f_{hv}\} & 2\mathrm{Im}\{f_{vh}\} & \mathrm{Im}\{f_{vv}+f_{hh}\} & \mathrm{Re}\{f_{vv}-f_{hh}\} \\ 2\mathrm{Re}\{f_{hv}\} & -2\mathrm{Re}\{f_{vh}\} & \mathrm{Re}\{f_{hh}-f_{vv}\} & \mathrm{Im}\{f_{vv}+f_{hh}\} \end{bmatrix} \tag{2.16}$$

Besides the scattering and absorption from the particles, if the medium background absorption is not ignorable, a quantity $\kappa_{ag}\overline{\overline{I}}$ need to be added on the right side of Eq. (2.20), where κ_{ag} is the total gaseous absorption coefficients of the medium due to all constituent molecules $\kappa_{ag} = \sum_i \kappa_{agi}$ [15].

From the phase matrix and extinction matrix, the absorption vector $\overline{\kappa}_a$ can be expressed as

$$\overline{\kappa}_a = \begin{bmatrix} \kappa_{e_{11}} - \int\limits_{4\pi} \left[P_{11}(\Theta) + P_{21}(\Theta)\right] d\Omega' \\ \kappa_{e_{22}} - \int\limits_{4\pi} \left[P_{12}(\Theta) + P_{22}(\Theta)\right] d\Omega' \\ 2\kappa_{e_{13}} + 2\kappa_{e_{23}} - 2\int\limits_{4\pi} \left[P_{13}(\Theta) + P_{23}(\Theta)\right] d\Omega' \\ -2\kappa_{e_{14}} - 2\kappa_{e_{24}} + 2\int\limits_{4\pi} \left[P_{14}(\Theta) + P_{24}(\Theta)\right] d\Omega' \end{bmatrix} \tag{2.17}$$

where $\kappa_{e_{ij}}$ and P_{ij} are the (ij)th element of the matrices $\overline{\overline{\kappa}}_e$ and $\overline{\overline{P}}$, $i, j = 1, 2, 3, 4$.

2.1.2.3 The Forms for Spherical Particles

For spherical particles, due to symmetry of the spheres the cross polarization terms $f_{vh}(\Theta)$ and $f_{hv}(\Theta)$ are zero. Proof is given in Appendix A (and can be seen in many other publications). Thus the Stokes matrix in Eq. (2.11) has the following simplified form:

$$\overline{\overline{L}}(\Theta) = \begin{bmatrix} |f_{vv}(\Theta)|^2 & 0 & 0 & 0 \\ 0 & |f_{hh}(\Theta)|^2 & 0 & 0 \\ 0 & 0 & \mathrm{Re}\{f_{vv}(\Theta) f_{hh}^*(\Theta)\} & -\mathrm{Im}\{f_{vv}(\Theta) f_{hh}^*(\Theta)\} \\ 0 & 0 & \mathrm{Im}\{f_{vv}(\Theta) f_{hh}^*(\Theta)\} & \mathrm{Re}\{f_{vv}(\Theta) f_{hh}^*(\Theta)\} \end{bmatrix} \tag{2.18}$$

Consequentially, the extinction matrix is diagonal:

$$\overline{\overline{\kappa}}_e = \kappa_e(\Theta) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

where

$$\begin{aligned} \kappa_e(\Theta) &= \tfrac{4\pi n_o}{k} \mathrm{Im}\{f_{vv}(\Theta)\} \\ &= \tfrac{4\pi n_o}{k} \mathrm{Im}\{f_{hh}(\Theta)\} \end{aligned} \tag{2.20}$$

and the absorption vector has the first two elements equal and the last two elements equal to zero:

$$\overline{\kappa}_a = \begin{bmatrix} \kappa_e - \int\limits_{4\pi} [P_{11}(\Theta) + P_{21}(\Theta)]\, d\Omega' \\ \kappa_e - \int\limits_{4\pi} [P_{12}(\Theta) + P_{22}(\Theta)]\, d\Omega' \\ 0 \\ 0 \end{bmatrix} \tag{2.21}$$

As formulated, under the assumption of the scattered fields being incoherent, the DRTE describes a phenomenological relation of wave propagation based on the concept of energy conservation at each point in space and each frequency. Energy conservation and reciprocity within the framework of phase matrix, extinction matrix and absorption vector are discussed in Tsang **et al.** [40, 17]. However, macroscopic wave interference phenomena, such as propagation path bending in media with an inhomogeneous refractive index, coherent superposition of scattered waves near medium boundary, or multiple coherent scattering are not modeled by the DRTE [15].

2.2 Stokes Matrix: Transformation and Symmetry

The Stokes matrix (and the resulting phase matrix) discussed in the previous section is defined in the particle-based coordinate system (Fig. A.1). Although the particle-based system has advantages in providing simple forms of the scattering amplitudes for particles with symmetry, it is

necessary for modeling stratified media to express the scattering amplitudes in the principal coordinate system (Fig. 2.2, taken from Fig. 3.1 in [43]), defined by the scattering and incident angles $(\theta_s, \phi_s; \theta_i, \phi_i)$.

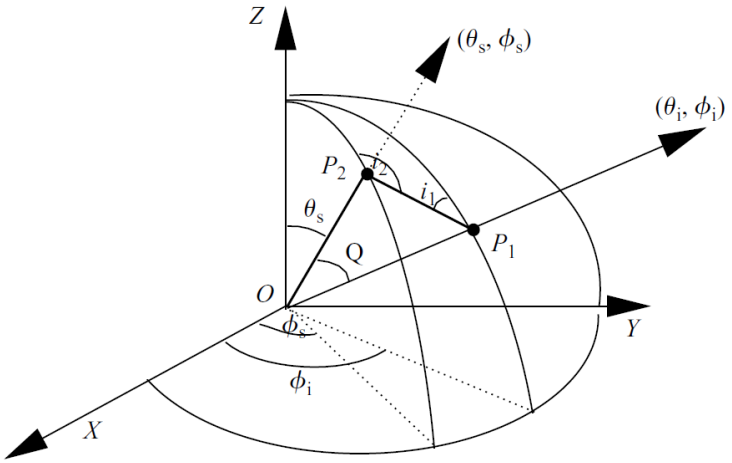


Figure 2.2: Geometry of the principal coordinate system. The incident intensity is defined by (θ_i, ϕ_i) and the scattered intensity is defined by (θ_s, ϕ_s).

The transformation of the Stokes matrix between the two coordinate systems is given by

$$\overline{\overline{L}}\left(\theta_s, \phi_s; \theta_i, \phi_i\right) = \overline{\overline{L}}_r\left(-i_2\right) \overline{\overline{L}}\left(\Theta\right) \overline{\overline{L}}_r\left(-i_1\right) \tag{2.22}$$

where $\overline{\overline{L}}_r$ is the rotation matrix [38, 39, 43]

$$\overline{\overline{L}}_r\left(i_{1,2}\right) = \begin{bmatrix} \cos^2 i_{1,2} & \sin^2 i_{1,2} & 0.5\sin 2i_{1,2} & 0 \\ \sin^2 i_{1,2} & \cos^2 i_{1,2} & -0.5\sin 2i_{1,2} & 0 \\ -\sin 2i_{1,2} & \sin 2i_{1,2} & \cos 2i_{1,2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

with following characteristics

$$\begin{aligned} \overline{\overline{L}}_r\left(i_1 + i_2\right) &= \overline{\overline{L}}_r\left(i_1\right) \overline{\overline{L}}_r\left(i_2\right) \\ \overline{\overline{L}}_r^{-1}\left(i_{1,2}\right) &= \overline{\overline{L}}_r\left(-i_{1,2}\right) \\ \overline{\overline{L}}_r\left(\pi\right) &= \overline{\overline{I}} \end{aligned} \tag{2.24}$$

and the angles i_1 and i_2 are the spherical surface angles defined between the planes OP_1Z and OP_1P_2, and OP_2Z and OP_1P_2, respectively. From [43], the cosine of these two angles are expressed as

$$\cos i_1 = \frac{\cos\theta_s \sin\theta_i - \cos\theta_i \sin\theta_s \cos\Delta\phi}{\sin\Theta} \tag{2.25a}$$

$$\cos i_2 = \frac{\cos\theta_i \sin\theta_s - \cos\theta_s \sin\theta_i \cos\Delta\phi}{\sin\Theta} \tag{2.25b}$$

where

$$\begin{aligned} \Delta\phi &= \phi_i - \phi_s \\ \sin\Theta &= \sqrt{1 - \cos^2\Theta} \\ \cos\Theta &= \cos\theta_s \cos\theta_i + \sin\theta_s \sin\theta_i \cos\Delta\phi \end{aligned}$$

The spherical surface angles i_1 and i_2 can be computed by following equations without ambiguity:

$$i_{1,2} = \begin{cases} 2\pi - \operatorname{acos}\left[\cos(i_{1,2})\right], & \pi < \Delta\phi < 2\pi \\ \operatorname{acos}\left[\cos(i_{1,2})\right], & 0 < \Delta\phi < \pi \end{cases} \tag{2.26}$$

In general, $\overline{\overline{L}}\left(\theta_s, \phi_s; \theta_i, \phi_i\right)$ is a full 4×4 matrix whereas $\overline{\overline{L}}\left(\Theta\right)$ has only six nonzero elements, four of which are independent.

The analytical diagonalization and factorization technique used within UMRT requires symmetry of the phase (and thus Stokes) matrix in the principal coordinate system under scattering path reversal (i.e.,$\theta_s \leftrightarrow \theta_i$). To show this degree of symmetry, the following equalities are examined by applying the coordinate transformation defined within (2.22-2.26):

$$\overline{\overline{L}}\left(\theta_s, \theta_i; \triangle\phi\right) \stackrel{?}{=} \left[\overline{\overline{L}}\left(\theta_i, \theta_s; \triangle\phi\right)\right]^T \tag{2.27a}$$

$$\overline{\overline{L}}\left(\theta_s, \pi - \theta_i; \triangle\phi\right) \stackrel{?}{=} \left[\overline{\overline{L}}\left(\theta_i, \pi - \theta_s; \triangle\phi\right)\right]^T \tag{2.27b}$$

for which they would (respectively) follow that:

$$\overline{\overline{L}}_r\left(-i_2\right)\overline{\overline{L}}\left(\Theta\right)\overline{\overline{L}}_r\left(-i_1\right) \stackrel{?}{=} \left[\overline{\overline{L}}_r\left(-i_1\right)\overline{\overline{L}}\left(\Theta\right)\overline{\overline{L}}_r\left(-i_2\right)\right]^T \tag{2.27c}$$

$$\overline{\overline{L}}_r\left(-i_1\right)\overline{\overline{L}}\left(\Theta\right)\overline{\overline{L}}_r\left(-i_2\right) \stackrel{?}{=} \left[\overline{\overline{L}}_r\left(i_2\right)\overline{\overline{L}}\left(\Theta\right)\overline{\overline{L}}_r\left(i_1\right)\right]^T \tag{2.27d}$$

Applying a Stokes matrix $\overline{\overline{L}}\left(\Theta\right)$ for a spherical particle with the form as in Eq. (2.18) to Eqs. (2.27c and 2.27d), the equalities in (2.27c-2.27d) hold for the diagonal and $v-h$ elements, viz:

$$\overline{\overline{\Delta}} = \begin{bmatrix} 0 & 0 & \Delta_{13} & \Delta_{14} \\ 0 & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{31} & \Delta_{32} & 0 & \Delta_{34} \\ \Delta_{41} & \Delta_{42} & \Delta_{43} & 0 \end{bmatrix} \tag{2.28}$$

where $\overline{\overline{\Delta}} \triangleq \overline{\overline{L}}\left(\theta_s, \theta_i; \triangle\phi\right) - \left[\overline{\overline{L}}\left(\theta_i, \theta_s; \triangle\phi\right)\right]^T$ or $\overline{\overline{L}}\left(\theta_s, \pi - \theta_i; \triangle\phi\right) - \left[\overline{\overline{L}}\left(\theta_i, \pi - \theta_s; \triangle\phi\right)\right]^T$ and Δ_{ij} represents a non-zero matrix element. Eq. (2.28) shows that the Stokes matrix $\overline{\overline{L}}\left(\theta_s, \theta_i; \triangle\phi\right)$ for spheres is symmetric for the first two Stokes' parameters. More specifically, if $\overline{\overline{L}}\left(\Theta\right)$ is calculated from either the Mie or DMRT scattering theory, the difference matrix $\overline{\overline{\Delta}}$ is found by numerical calculation for a wide range of parameters and angles comprising nearly one million diverse cases to be:

$$\overline{\overline{\Delta}} = \begin{bmatrix} \sim 0 & \sim 0 & \sim 0 & \sim 0 \\ \sim 0 & \sim 0 & \sim 0 & \sim 0 \\ \sim 0 & \sim 0 & \sim 0 & \Delta_{34} \\ \sim 0 & \sim 0 & \Delta_{43} & \sim 0 \end{bmatrix} \tag{2.29}$$

where the $' \sim 0'$ entries are zero within standard **IEEE** numerical precision. While not an absolute proof the above strongly suggests that both the Mie and DMRT Stokes matrices are symmetric for the first three Stokes' parameters.

Moreover, if the Stokes matrix $\overline{\overline{L}}(\Theta)$ has the simplified form of the Rayleigh Stokes matrix (i.e., for electrically small particles):

$$\overline{\overline{L}}(\Theta) = \frac{3}{2}\begin{bmatrix} \cos^2\Theta & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & \cos\Theta \end{bmatrix} \tag{2.30}$$

Then it can be shown that $\overline{\overline{\Delta}} = \overline{\overline{0}}$ for all entries. In this case of small particles $\overline{\overline{L}}(\theta_s, \theta_i; \triangle\phi)$ is symmetric for all four Stokes' parameters.

2.3 Phase Matrix Used in Unified Microwave Radiative Transfer Model

2.3.1 Planar-Stratified Medium Approximation

2.3.1.1 Reduced Phase Matrix

Planar-stratified medium approximation is applied in UMRT. For example, a planar-stratified sea ice model is illustrated in Fig. 2.3. In such model, the layer of air is categorized as sparse medium, other layers are categorized as dense medium layers (i.e., snow and sea ice) and sea water is treated as the background.

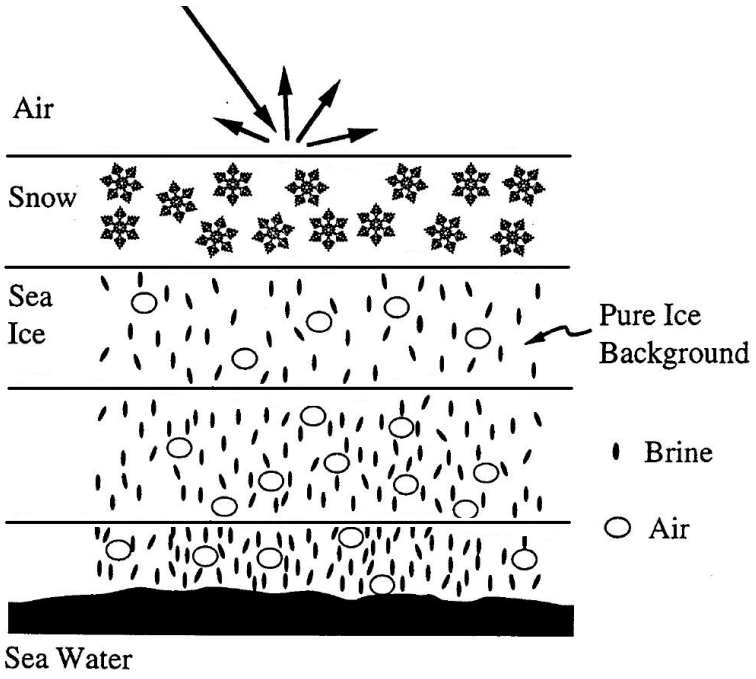


Figure 2.3: A sea ice model with planar-stratified medium structure.

Due to the azimuthal symmetry of the planar-stratified approximation, only the vertical and horizontal polarized Stokes' parameters are required to describe the brightness temperature field and thus, the DRTE in Eq. (2.2) is simplified as

$$\begin{aligned}\cos\theta_s \frac{d\overline{T}_B(h,\theta_s,f)}{dh} = & -\overline{\overline{\kappa}}_e(h,\theta_s,f)\cdot\overline{T}_B(h,\theta_s,f) + \overline{\kappa}_a(h,\theta_s,f)\cdot\overline{T}(h) \\ & +\int_0^{\pi}\overline{\overline{P}}'(h,\theta_s,\theta_i,f)\cdot\overline{T}_B(h,\pi-\theta_i,f)\cdot\sin\theta_i d\theta_i\end{aligned} \tag{2.31}$$

where h is the thickness of layer and $\overline{\overline{P}}'(h,\theta_s,\theta_i,f)$ is the reduced (2×2) form of the general (4×4) phase matrix: [15]

$$\overline{\overline{P}}'(h,\theta_s,\theta_i,f) \equiv \int_0^{2\pi}\overline{\overline{P}}(h,\theta_s,\theta_i,f;\triangle\phi)\,d(\triangle\phi) \tag{2.32}$$

where

$$\overline{\overline{P}}(h,\theta_s,\theta_i,f;\triangle\phi) = \int_0^{\infty}\overline{\overline{L}}(\theta_s,\theta_i;\triangle\phi)\cdot n(D)\,dD$$

the reduced phase matrix is only a function of the incident angle θ_i and scattering angle θ_s and the matrix elements $P'_{\alpha\beta}$ represents the α-polarized scattered (θ_s) radiation ($\alpha = v, h$) due to the β-polarized incident (θ_i) radiation ($\beta = v, h$).

Due to the azimuthal symmetry within the Mie and DMRT theories it can be shown that the reduced phase matrix becomes:

$$\overline{\overline{P}}'(\theta_s, \theta_i)_{\text{Mie/DMRT}} = \begin{bmatrix} P_{11} & P_{12} & 0 & 0 \\ P_{21} & P_{22} & 0 & 0 \\ 0 & 0 & P_{33} & P_{34} \\ 0 & 0 & P_{43} & P_{44} \end{bmatrix} \tag{2.33}$$

where it is seen that the 1^{st} and 2^{nd} Stokes' parameters are decoupled from the $3^{\text{rd}}$and 4^{th}.

Particularly, the reduced Rayleigh phase matrix is

$$\overline{\overline{P}}'(\theta_s, \theta_i)_{\text{Rayleigh}} = \begin{bmatrix} P_{11} & P_{12} & 0 & 0 \\ P_{21} & P_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{44} \end{bmatrix} \tag{2.34}$$

where the first three Stokes' parameters are decoupled from the 4^{th}.

For the spherical particle case and the later Jacobian calculation, it is convenient to define the reduced normalized phase matrix [15]:

$$\overline{\overline{p}}'(\theta_s, \theta_i) \equiv \frac{\overline{\overline{P}}'(\theta_s, \theta_i)}{\kappa_s} \tag{2.35}$$

where $\int_0^{\pi} \overline{\overline{p}}'(\theta_s, \theta_i) \sin\theta_s d\theta_s = 1$.

2.3.1.2 Reduced Extinction Matrix and Absorption Vector

In the planar-stratified case, the extinction matrix and absorption vector in Eqs. (2.20) and (2.21) are consequently reduced to the following simplified forms:

$$\begin{aligned} \overline{\overline{\kappa}}_e(h,f) &= \kappa_e \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \overline{\kappa}_a(h,f) &= (\kappa_e - \kappa_s) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned} \tag{2.36}$$

where

$$\begin{aligned} \kappa_e &= \tfrac{4\pi n_o}{k} \langle \mathrm{Im}\{f_{vv}(\Theta)\} \rangle \\ &= \tfrac{4\pi n_o}{k} \langle \mathrm{Im}\{f_{hh}(\Theta)\} \rangle \end{aligned} \tag{2.37}$$

$$\begin{aligned} \kappa_s &= \int_{4\pi} [P_{11}(\Theta) + P_{21}(\Theta)]\, d\Omega' \\ &= \int_{4\pi} [P_{12}(\Theta) + P_{22}(\Theta)]\, d\Omega' \end{aligned} \tag{2.38}$$

where κ_s is the scattering coefficient and background gaseous absorption is not included in κ_e.

2.3.1.3 Fresnel Relations

For a plane wave incident from medium 1 (ϵ_1) to medium 2 (ϵ_2) at the specular interface, the reflectivity is calculated using the Fresnel relations:

$$r_v = |R_v|^2 = \left| \frac{\frac{\epsilon_2}{\epsilon_1}\cos\theta_i - \sqrt{\frac{\epsilon_2}{\epsilon_1} - \sin^2\theta_i}}{\frac{\epsilon_2}{\epsilon_1}\cos\theta_i + \sqrt{\frac{\epsilon_2}{\epsilon_1} - \sin^2\theta_i}} \right|^2 \tag{2.39}$$

for vertically polarized waves and

$$r_h = |R_h|^2 = \left| \frac{\cos\theta_i - \sqrt{\frac{\epsilon_2}{\epsilon_1} - \sin^2\theta_i}}{\cos\theta_i + \sqrt{\frac{\epsilon_2}{\epsilon_1} - \sin^2\theta_i}} \right|^2 \tag{2.40}$$

for horizontally polarized waves. In above equations permeabilities of both medium are assumed to be $\mu_1 = \mu_2 = 1$, and the Fresnel transmissivity is one minus the reflectivity by energy conservation.

Permittivity of natural medium is important in microwave radiometry in several ways: 1) the wave impedance and the scattering properties are related to the permittivity, 2) the absorption properties directly follow from the imaginary part of permittivity of the propagating medium, and

3) the ray path of the radiation (determined by Snell's law) depends on variation in the permittivity. Although it is beyond the scope of this thesis, several commonly used (in microwave radiometry) permittivity models for liquid pure water and sea water, dry snow, wet snow, and sea ice are introduced in Appendix B. More detailed information on this topic can be found in the literature on electromagnetic waves and fields.

2.3.2 Rayleigh Phase Matrix

For small particle size parameters ($ka < 0.1$) the reduced normalized Rayleigh phase matrix is

$$\bar{\bar{p}}'\left(\theta_s, \theta_i\right)_{\text{Rayleigh}} = \frac{3}{8}\begin{bmatrix} 2\sin^2\theta_s \sin^2\theta_i + \cos^2\theta_s \cos^2\theta_i & \cos^2\theta_s \\ \cos^2\theta_i & 1 \end{bmatrix} \tag{2.41}$$

where k is the wavenumber in air and a is the sphere radius.

It is seen from Eq. (A.6) that the reduced normalized Rayleigh phase matrix is independent of frequency. An example is given in Fig. 2.4, from where it can be seen that P_{11} is symmetric in angles, P_{12} is a 90^{o} rotation of P_{21}, and P_{22} is constant. These behaviors are consistent with the symmetry conclusion of §2.2 and the above Rayleigh phase matrix expression.

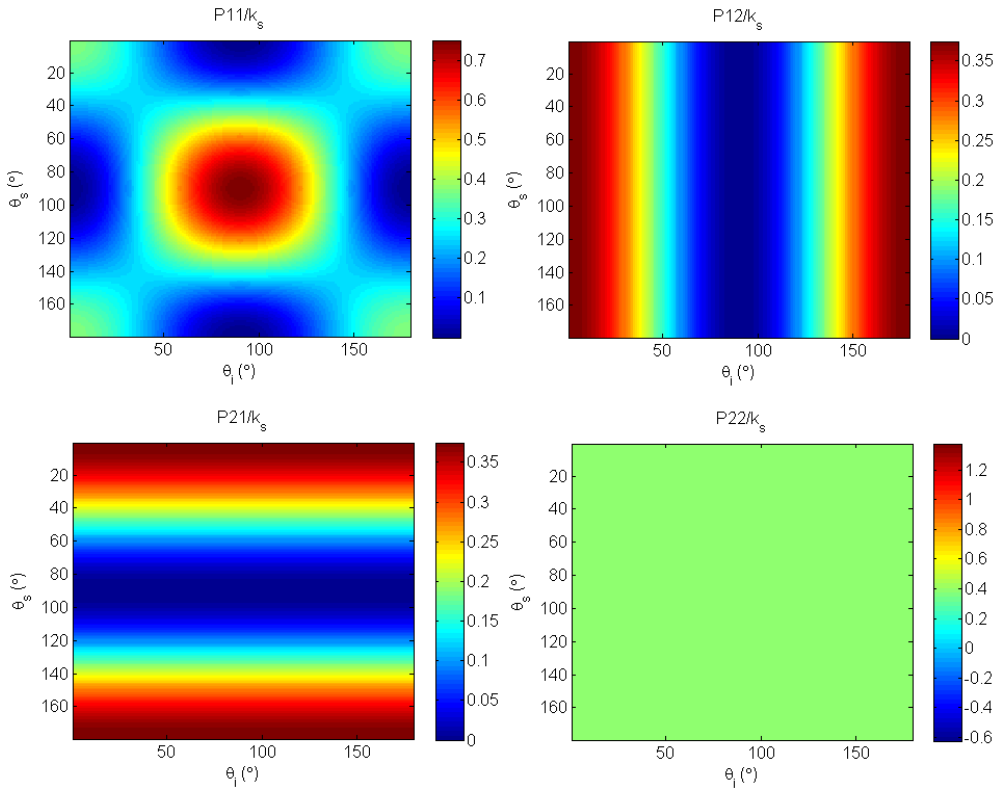


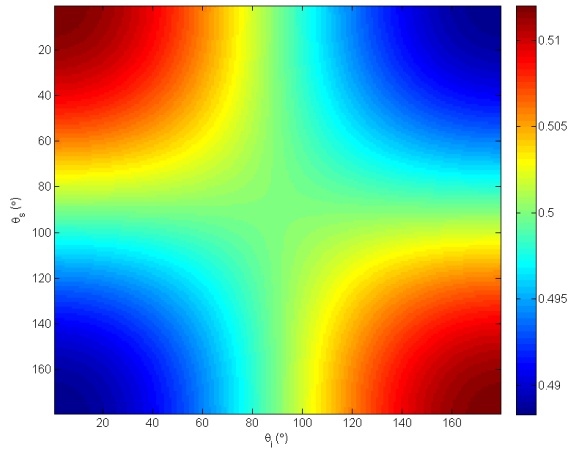
Figure 2.4: Reduced normalized Rayleigh phase matrix. Both θ_s and θ_i are equally discretized by 179 steps.

2.3.3 Henyey-Greenstein Phase Matrix Approximation

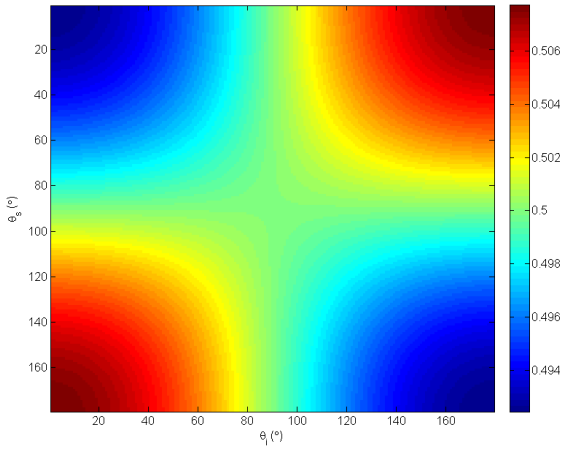
The Henyey-Greenstein (HG) phase matrix approximation is used in DOTLRT. It takes anisotropic scattering into account through the polydispersed Mie asymmetry parameter g (c.f. §2.3.4), and is positive definite. It has the following matrix form:

$$\overline{\overline{p}}'\left(\theta_s,\theta_i\right)_{\mathrm{HG}} = \frac{1-g^2}{2\pi}\int_0^{\pi}\frac{d\left(\triangle\phi\right)}{\left[1+g^2-2g\cos\theta_s\cos\theta_i+2g\sin\theta_s\sin\theta_i\cos\triangle\phi\right]^{3/2}}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{2.42}$$

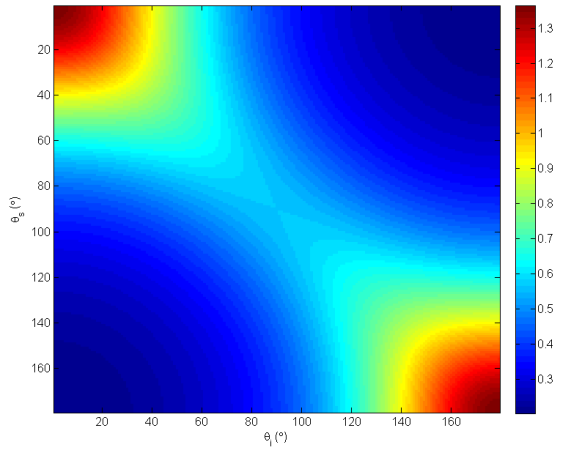
By varying the asymmetry parameter g, the HG phase matrix can model scattering in forward-, isotropic-, backward-, and any degree in between these extremes. According to [15] the asymmetry parameter g is positive (indicating predominantly forward scattering) for ice; however it is negative for liquid over a small but important frequency range (~10-30GHz). Fig. 2.5 shows examples of



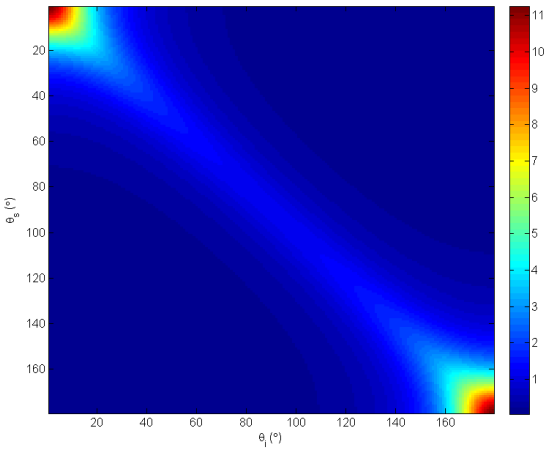
(a) 10 GHz



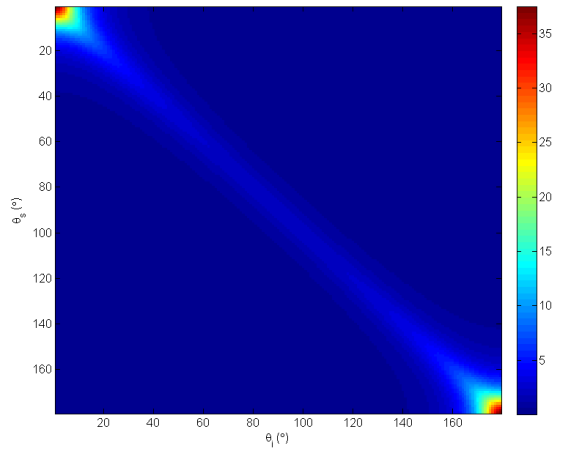
(b) 30 GHz



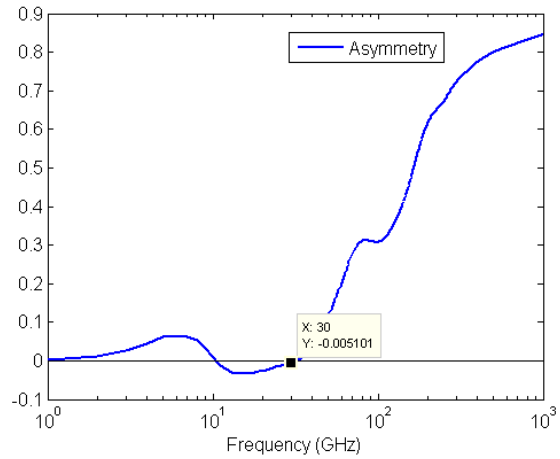
(c) 100 GHz



(d) 300 GHz



(e) 1000 GHz



(f) The polydispersed Mie asymmetric coefficient

Figure 2.5: Reduced normalized Henyey-Greenstein phase matrix (a-e), and the polydispersed Mie asymmetric coefficient (f). Both θ_s and θ_i are equally discretized by 179 steps.

the HG phase matrix at frequencies: 10, 30, 100, 300, and 1000 GHz for a rain case assuming mean drop diameter $\langle D \rangle = 2$ mm and applying the Marshall-Palmer (MP) particle size distribution with a 10 mm/hr rain rate. It is seen that the HG phase matrix is predominantly forward scattering as frequency increases except for the frequency at 30 GHz (Fig. 2.5(b)), which matches the negative behavior of the Mie polydispersed asymmetry coefficient g, shown in Fig. 2.5(f).

2.3.4 Mie Phase Matrix

The equations related to the calculation of full Mie phase matrix are summarized here by first noting that the Mie scattering amplitudes are [45, 46]:

$$f_{vv}(\Theta) = \frac{-j}{k} \sum_{n=1}^{n_{max}} \frac{2n+1}{n(n+1)} \left[a_n \pi_n (\cos\Theta) + b_n \tau_n (\cos\Theta) \right] \tag{2.43a}$$

$$f_{hh}(\Theta) = \frac{-j}{k} \sum_{n=1}^{n_{max}} \frac{2n+1}{n(n+1)} \left[a_n \tau_n (\cos\Theta) + b_n \pi_n (\cos\Theta) \right] \tag{2.43b}$$

where k is the wavenumber in air, (a_n, b_n) are the Mie scattering coefficients

$$a_n = -\frac{j_n(mx)\left[x j_n(x)\right]' - j_n(x)\left[mx j_n(mx)\right]'}{j_n(mx)\left[x h_n(x)\right]' - h_n(x)\left[mx j_n(mx)\right]'} \tag{2.44a}$$

$$b_n = -\frac{j_n(x)\left[mx j_n(mx)\right]' - m^2 j_n(mx)\left[x j_n(x)\right]'}{h_n(mx)\left[mx j_n(mx)\right]' - m^2 j_n(mx)\left[x h_n(x)\right]'} \tag{2.44b}$$

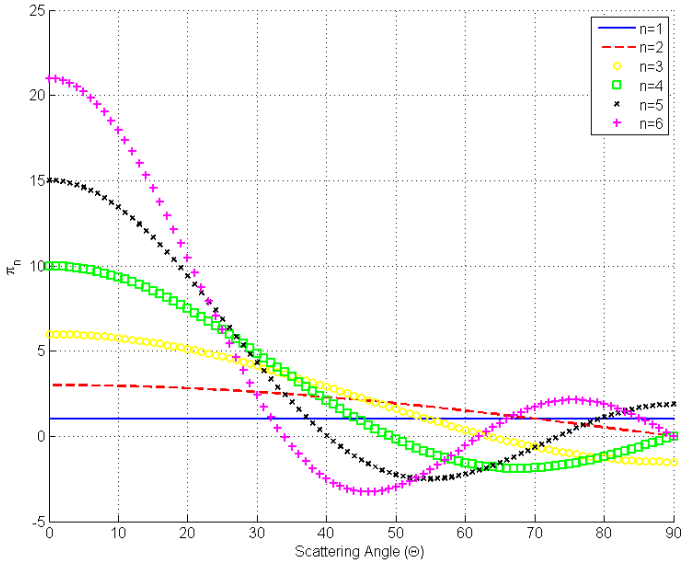
and (π_n, τ_n) are the angle-dependent functions, which are defined in a upward recurrence

$$\pi_n = \frac{2n-1}{n-1} \cos\Theta \cdot \pi_{n-1} - \frac{n}{n-1} \pi_{n-2} \tag{2.45a}$$

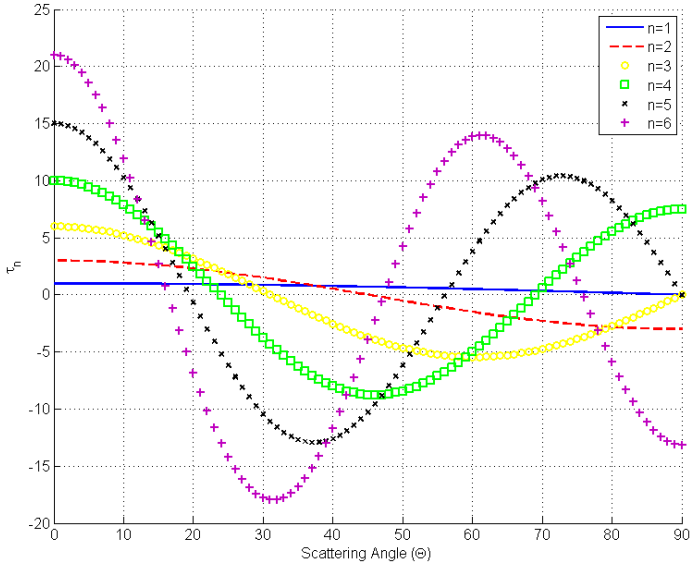
$$\tau_n = n \cos\Theta \cdot \pi_n - (n+1)\pi_{n-1} \tag{2.45b}$$

where $\pi_0 = 0$ and $\pi_1 = 1$. The behaviors of (π_n, τ_n) for n up to 6 are shown in Fig. 2.6.

The choice of maximum iteration number is commonly determined by $n_{max} = \text{round}\left(x + 4x^{\frac{1}{3}} + 2\right)$, where $x = ka$ is the size parameter and the operation, round$(\cdot)$ returns the closest integer less than $(\cdot)$.



(a) π_n



(b) τ_n

Figure 2.6: The angle-dependent functions (π_n, τ_n) depicted as functions of forward scattering angle Θ and for an n up to 6.

Accordingly, the Mie phase matrix elements under a specific particle size distribution function $n(D)$ are computed as:

$$\overline{\overline{P}}(\Theta) = \begin{bmatrix} P_{11}(\Theta) & 0 & 0 & 0 \\ 0 & P_{22}(\Theta) & 0 & 0 \\ 0 & 0 & P_{33}(\Theta) & P_{34}(\Theta) \\ 0 & 0 & P_{43}(\Theta) & P_{44}(\Theta) \end{bmatrix} \tag{2.46a}$$

where

$$P_{11}(\Theta) = \int_0^\infty |f_{vv}(\Theta)|^2 \cdot n(D)\, dD \tag{2.46b}$$

$$P_{22}(\Theta) = \int_0^\infty |f_{hh}(\Theta)|^2 \cdot n(D)\, dD \tag{2.46c}$$

$$P_{33}(\Theta) = \int_0^\infty \mathrm{Re}\{f_{vv}(\Theta) \cdot f_{hh}^*(\Theta)\} \cdot n(D)\, dD \tag{2.46d}$$

$$P_{34}(\Theta) = -\int_0^\infty \mathrm{Im}\{f_{vv}(\Theta) \cdot f_{hh}^*(\Theta)\} \cdot n(D)\, dD \tag{2.46e}$$

$$P_{43}(\Theta) = -P_{34}(\Theta), \quad P_{44}(\Theta) = P_{33}(\Theta) \tag{2.46f}$$

To compute the reduced Mie phase matrix, it is usually convenient to integrate the above expressions numerically with respect to the azimuthal angle, as in Eq. (2.32).

Within UMRT, for sparse media the extinction and scattering coefficients κ_e and κ_s are calculated based on the Mie theory for polydispersed particles [15] while for dense media they are calculated differently under the DMRT-QCA theory (c.f. §2.3.4). Using Mie theory the efficiencies η_e and η_s, and the fraction of power G for monodispersed spherical particles are computed as:

$$\eta_e = \frac{2}{x^2} \sum_{n=1}^{n_{max}} (2n+1)\, \mathrm{Re}(a_n + b_n) \tag{2.47a}$$

$$\eta_s = \frac{2}{x^2}\sum_{n=1}^{n_{max}} (2n+1)\left(|a_n|^2 + |b_n|^2\right) \tag{2.47b}$$

$$G = \frac{1}{\eta_s}\frac{4}{x^2}\sum_{n=1}^{n_{max}}\left[\frac{n\,(n+2)}{(n+1)}\mathrm{Re}\left\{a_n^* a_{n+1} + b_n^* b_{n+1}\right\} + \frac{2n+1}{n(n+1)}\mathrm{Re}\left\{a_n b_n^*\right\}\right] \tag{2.47c}$$

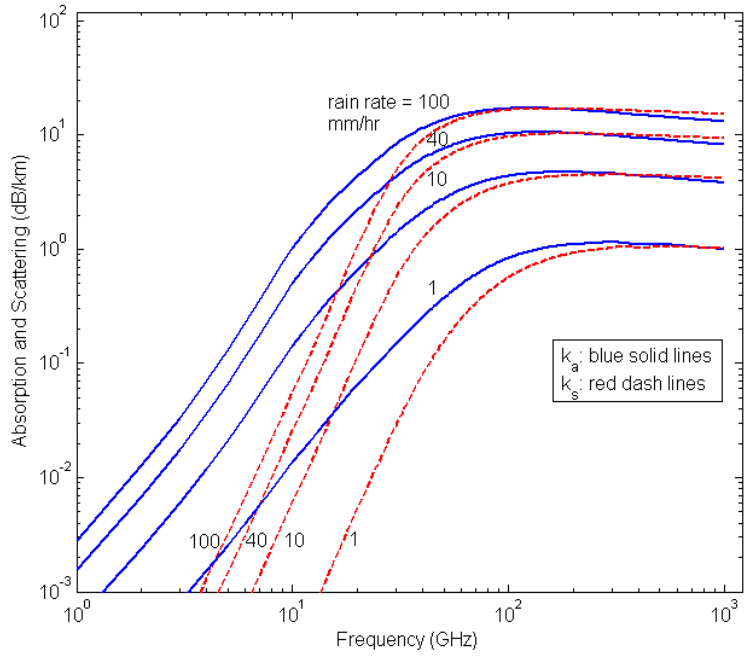
Given a particle size distribution function $n(D)$ the coefficients for polydispersed spherical particles are computed as:

$$\kappa_e = \frac{\pi}{4}\int_0^\infty \eta_e \cdot D^2 \cdot n\,(D)\,dD \tag{2.48a}$$

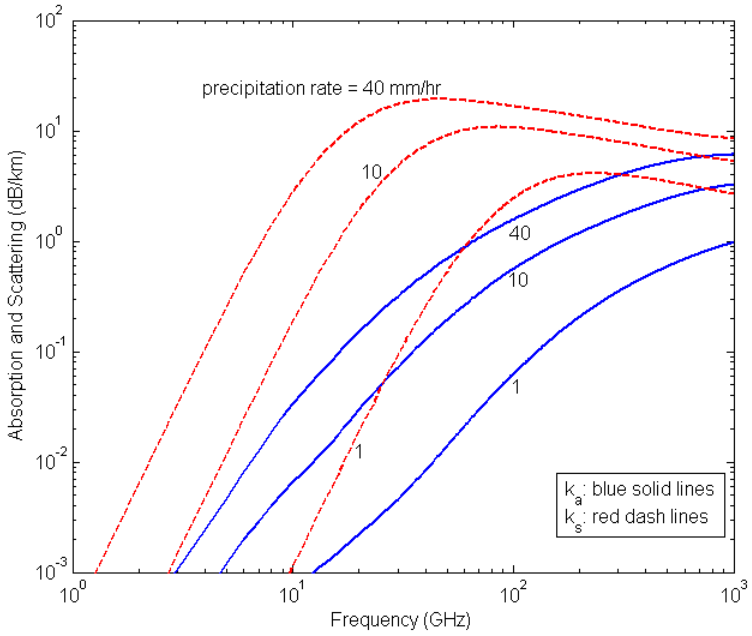
$$\kappa_s = \frac{\pi}{4}\int_0^\infty \eta_s \cdot D^2 \cdot n\,(D)\,dD \tag{2.48b}$$

$$g = \frac{\int_0^\infty G \cdot \eta_s \cdot D^2 \cdot n\,(D)\,dD}{\int_0^\infty \eta_s \cdot D^2 \cdot n\,(D)\,dD} \tag{2.48c}$$

Following [15] the upper limit of the above integrations are set to be 15 times the mean particle diameter $\langle D \rangle$. Since $n(D)$ is typically an exponential function integrand contributions typically diminish after a few mean diameters. Figs. 2.7a-b shows examples of theoretical calculations of the polydispersive Mie scattering and absorption coefficients for both liquid and ice spheres for various precipitation rates at 0 °C. The two figures are perfectly reproduced to match the figures published in chapter 3, [15] as a part of the UMRT program validation procedure. Examples of the Mie phase matrix will be provided in §2.3.6.



(a)



(b)

Figure 2.7: The scattering and absorption coefficients for polydispersive Mie spherical particles. (a) Liquid, assuming a Marshall-Palmer particle size distribution. (b) Ice, assuming a Sekhon-Srivastava distribution. Calculations are shown for precipitation rates of 1, 10 and 40 mm/hr for both phases and 100 mm/hr for liquid.

2.3.5 DMRT-QCA Phase Matrix

UMRT employs the DMRT-QCA model outlined in [37], which simplifies the calculation of the DMRT-QCA phase matrix relative to previous implementations. The effective propagation constant K and the average multipole amplitudes $X_v^{(M)}$ and $X_v^{(N)}$ are numerically calculated by solving the $2N_{max}$ system of equations obtained using the Lorentz-Lorentz (L-L) law and the Ewald-Oseen extinction theorem. The above quantities are subsequently used to calculate the DMRT-QCA Stokes matrix elements:

$$\begin{aligned} f_{vv}(\Theta) = & \frac{-j}{(1-R)}\sqrt{\frac{1}{kK_r}}\sum_{n=1}^{N_{max}}\frac{2n+1}{n(n+1)} \\ & \times\left[a_n X_n^{(N)}\pi_n(\cos\Theta)+b_n X_n^{(M)}\tau_n(\cos\Theta)\right] \\ f_{hh}(\Theta) = & \frac{-j}{(1-R)}\sqrt{\frac{1}{kK_r}}\sum_{n=1}^{N_{max}}\frac{2n+1}{n(n+1)} \\ & \times\left[a_n X_n^{(N)}\tau_n(\cos\Theta)+b_n X_n^{(M)}\pi_n(\cos\Theta)\right] \end{aligned} \tag{2.49a}$$

where k is the wavenumber in air, $K_r = \mathrm{Re}\{K\}$, and R is a coefficient

$$R = \frac{-j\pi n_o}{k^2(k+K_r)}\sum_{n=1}^{N_{max}}(-1)^n\left[-b_n X_n^{(M)}+a_n X_n^{(N)}\right](2n+1) \tag{2.49b}$$

where $n_o = \frac{6f_v}{8\pi a^3}$ is the particle number density, where f_v is volume fraction of the particle.

The phase matrix elements are:

$$P_{11}(\Theta) = |f_{vv}(\Theta)|^2 q(\Theta) \tag{2.50a}$$

$$P_{22}(\Theta) = |f_{hh}(\Theta)|^2 q(\Theta) \tag{2.50b}$$

$$P_{33}(\Theta) = P_{44}(\Theta) = \mathrm{Re}\{f_{vv}(\Theta)\cdot f_{hh}^*(\Theta)\} q(\Theta) \tag{2.50c}$$

$$P_{34}(\Theta) = -P_{43}(\Theta) = -\mathrm{Im}\{f_{vv}(\Theta)\cdot f_{hh}^*(\Theta)\} q(\Theta) \tag{2.50d}$$

where the factor $q(\Theta)$ is obtained using the Percus-Yevick (PY) approximation in ([37], eqs. 10-11).

In DMRT-QCA, the scattering and absorption coefficients are computed as follows:

$$\kappa_a = \frac{k}{K_r}\frac{2\pi}{k^2|1-R|^2} n_o \cdot \sum_{n=1}^{N_{max}} (2n+1)[\left|X_n^{(M)}\right|^2 \cdot \left(\mathrm{Re}\{b_n\} - |b_n|^2\right) + \left|X_n^{(N)}\right|^2 \left(\mathrm{Re}\{a_n\} - |a_n|^2\right)] \tag{2.51a}$$

$$\kappa_s = \pi \int_0^{\infty} [P_{11}(\Theta) + P_{22}(\Theta)] \sin\Theta d\Theta \tag{2.51b}$$

$$\kappa_e = \kappa_a + \kappa_s \tag{2.51c}$$

An example of the DMRT-QCA scattering coefficient is shown in Fig. 2.8 using the following conditions listed in [47]: the particle mean diameter $\langle D \rangle$ is 1.2 mm, the volume fraction f_v is 30%, the permittivity of the particles is $\epsilon_s = 3.2\epsilon_o$, and the stickiness parameter τ is 0.1. As seen in Fig. 2.8, the UMRT calculations compare favorably with the calculations by Tsang **et al.** [47].

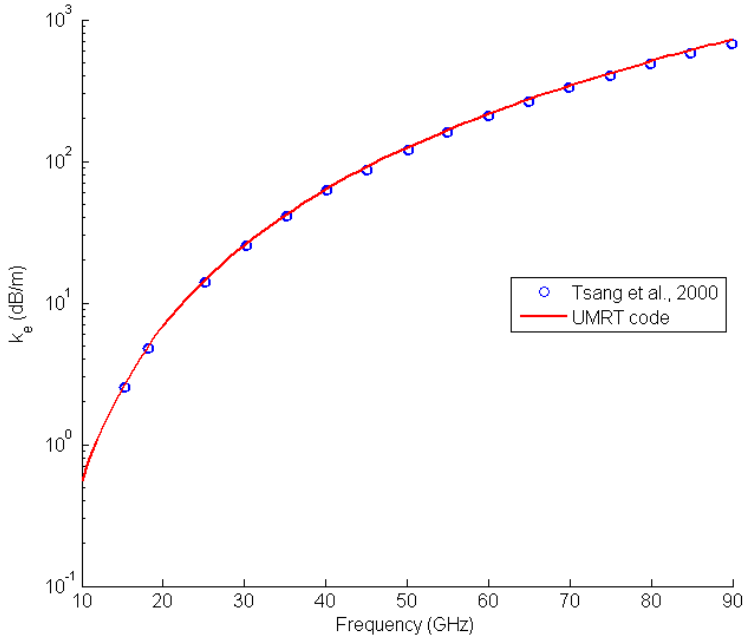


Figure 2.8: Comparison of extinction rate as a function of frequency for sticky DMRT-QCA model

As a short summary of this section, the procedure of computing the Mie and DMRT-QCA

phase matrices is illustrated in Fig. 2.9.

2.4 Results and Discussion

Comparisons of the scattering and absorption coefficients from the Mie and DMRT-QCA theories as functions of frequency for several typical conditions illustrate fundamental differences between these distinct models (Tab. 2.1 and Fig. 2.10). For purposes of comparison cases 2-6 use the fixed lossy value of ice permittivity of $\epsilon_{ice} = 3.15 - j0.001$, while case 1 uses the frequency dependent value obtained from [48].

Table 2.1: Conditions for scattering and absorption coefficient calculations for various ice particle distributions and for (1-2) DMRT-QCA theory, and (3-6) Mie theory, for which cases 3-4 use sparse Sekhon-Srivastava (SS) ice particle size distributions for two nominal precipitation rates while 5-6 use dense exponential particle size distributions for a fixed volume fraction volume and two particle diameters.

	f_v	n_o ($\mathrm{m^{-3}\,mm^{-1}}$)	$\langle D \rangle$ (mm)
1) DMRT-QCA	2.5×10^{-1}	1.74×10^{2}	1.40
2) DMRT-QCA	2.5×10^{-1}	1.74×10^{2}	1.40
3) Mie, SS (PR = 10 mm/hr)	7.33×10^{-7}	2.87×10^{2}	1.23
4) Mie, SS (PR = 40 mm/hr)	1.07×10^{-7}	7.80×10^{1}	2.30
5) Mie, Dense exponential	2.5×10^{-1}	2.07×10^{7}	0.14
6) Mie, Dense exponential	2.5×10^{-1}	2.07×10^{3}	1.40

The differences in both κ_s and κ_a between the Mie and DMRT-QCA theories for identical ice volume fractions are seen in cases 1-2 and 5-6 of Fig. 2.10(a-b), where DMRT-QCA generally predicts smaller values for κ_s than the Mie theory for the same mean particle sizes and overall densities (cases 1-2 and 6). However, since absorption is more closely related to the internal field amplitude and particle volume the differences are smaller than for scattering. This effect is seen

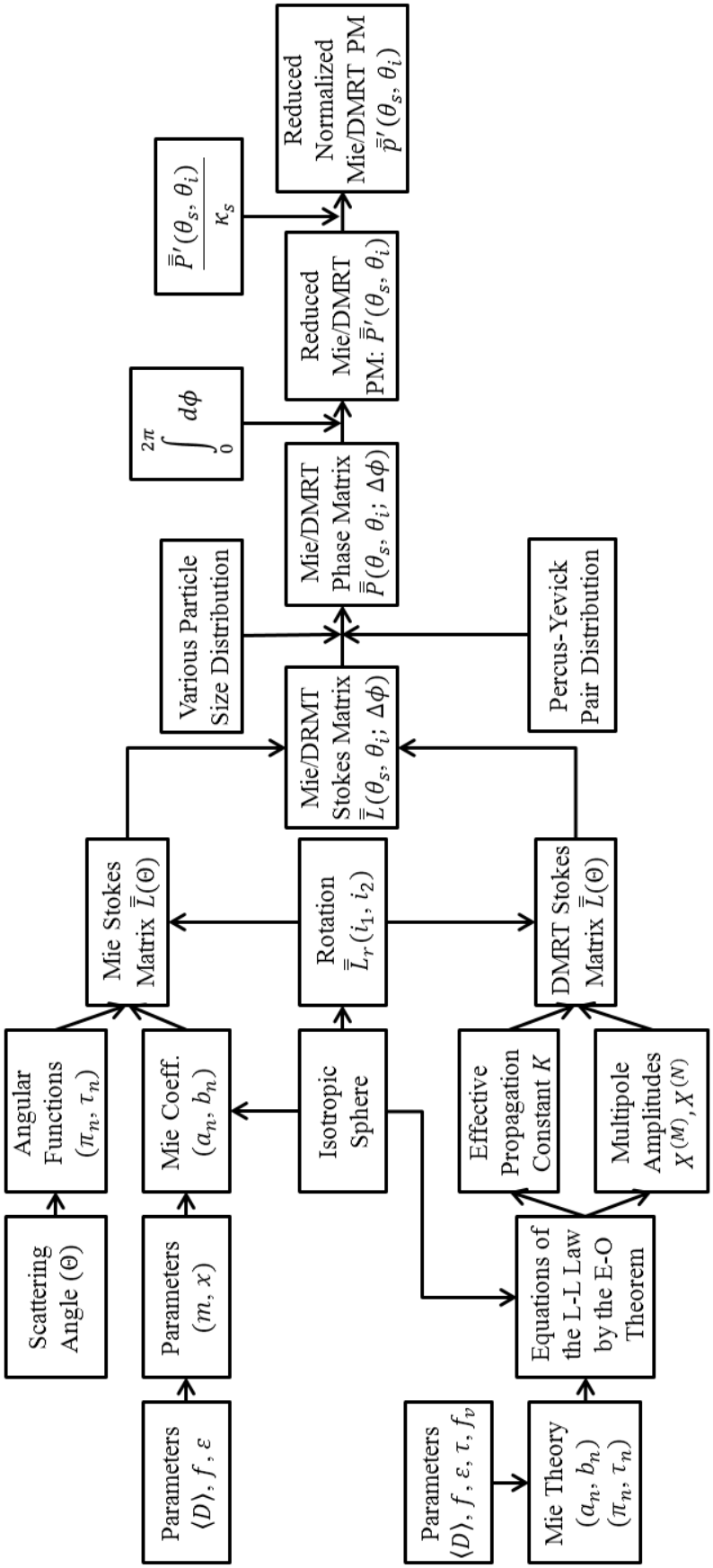
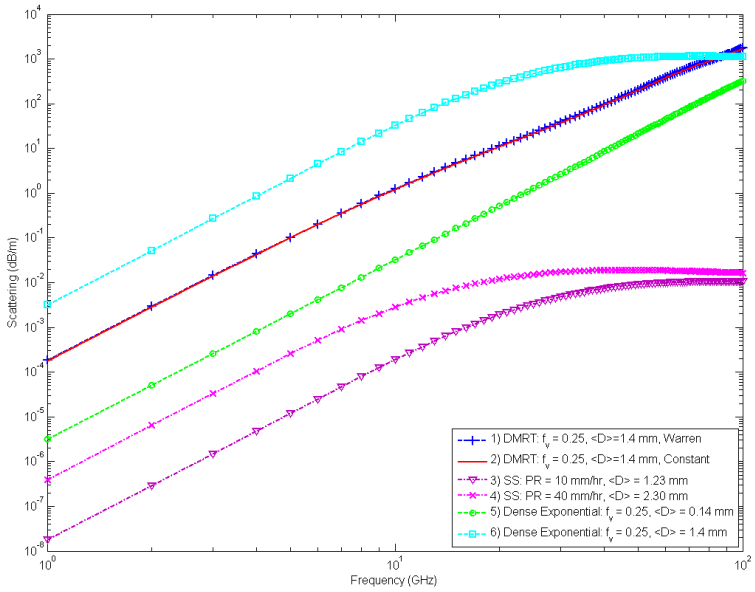
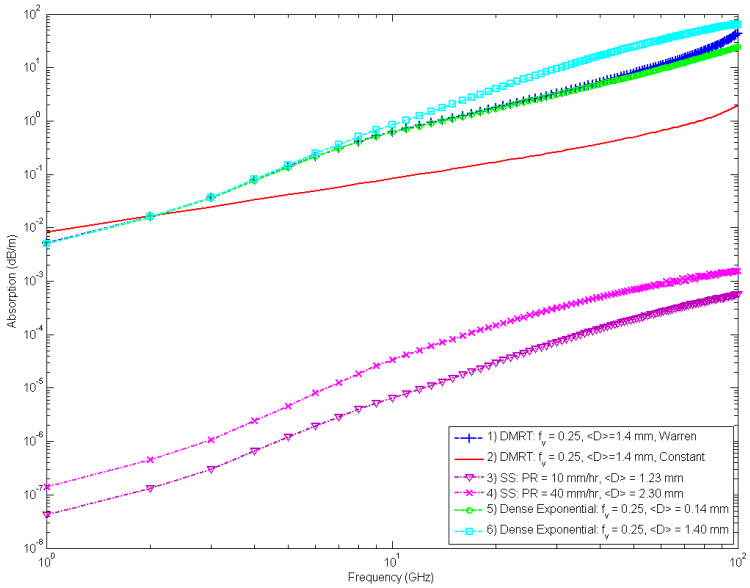


Figure 2.9: Block diagram of the calculation of full Mie and DMRT-QCA phase matrices.

more clearly by considering cases 5-6, where the mean particle size of the Mie simulation is varied by a factor of 10. For these cases the scattering coefficient for Rayleigh-sized particles increases by $\langle D \rangle^6 / n_o$, but there is less effect on the absorption coefficient, especially for the frequencies less than 10 GHz. In Fig. 2.10(a), we also note that κ_s of Mie theory saturates with



(a) Scattering Coefficients κ_s



(b) Absorption Coefficient κ_a

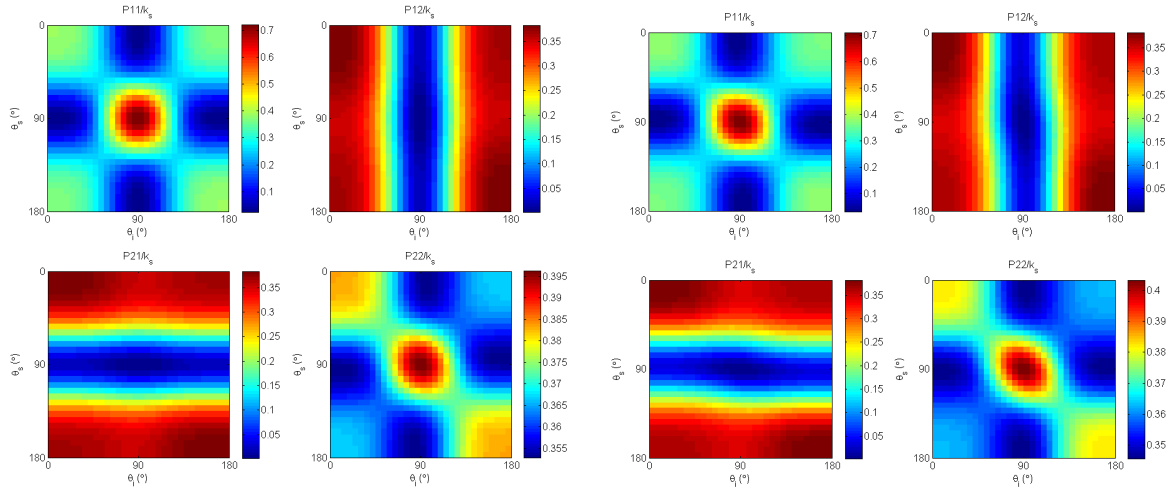
Figure 2.10: (a) Scattering and (b) absorption coefficients for polydispersive ice particles computed using Mie and DMRT-QCA theories.

larger particles at higher frequencies. This behavior suggests that the Mie scattering coefficient has a weaker frequency dependence than that of DMRT-QCA. As can be expected, Fig. 2.10(a-b) also show that the values of both κ_s and κ_a under Mie scattering for a dense distribution (cases 5-6) are much greater than their corresponding counterparts determined using the sparse Sekhon-Srivastava (SS, [29]) distribution (cases 3-4). This difference is the result of scaling by the volume fraction, and is inherent in Mie theory. However, the DMRT-QCA scattering coefficient depends non-linearly on f_v, and is accurately computable to volume fractions of at least ~20% [49]. Finally, in cases 1-2 it is noted that use of the nominal value for the ice dielectric constant in computing the value of κ_s does not result in obvious differences when compared with results using the ice dielectric constant values from Warren [48], however, these two dielectric constant models do result in significant differences in the value of κ_a. Accordingly, improved models of the dielectric constant of homogeneous water ice are suggested to be of interest.

The behavior of reduced normalized Mie phase matrices are studied by assuming a rain case with following conditions: 1) Marshall-Palmer (MP) size distribution [23] with precipitation rate = 10 mm/hour and 2) mean drop diameter $\langle D \rangle = 2$ mm. The water dielectric constant is determined using the double Debye model [50] at a temperature of 0 °C. As seen in Fig. 2.11, the reduced normalized Mie phase matrices exhibit the expected symmetry for both vertical and horizontal polarizations. The plot further show that forward scattering relative to back- or side-scattering increases as frequency increases, gradually becoming dominant above ~100 GHz as suggested by calculations of polydispersive asymmetry in [15].

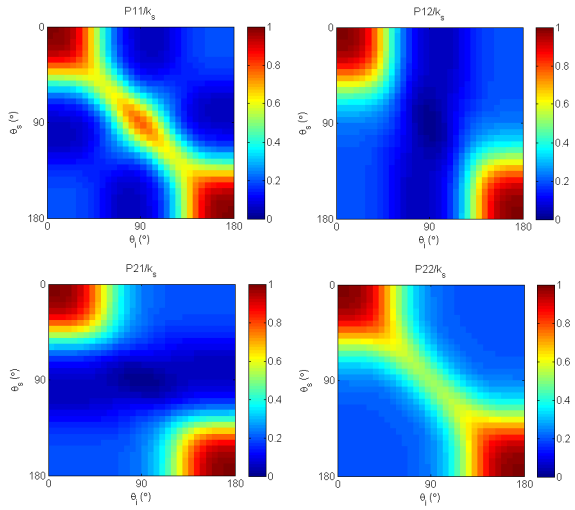
Analogously, Fig. 2.12 shows reduced normalized DMRT-QCA phase matrices computed for dense snowpack under the following conditions: 1) dielectric constant of ice of $\epsilon_{ice} = 3.15 - j0.001$, 2) mean ice diameter of $\langle D \rangle = 1.4$ mm, 3) volume fraction $f_v = 25\%$, 4) stickiness parameter $\tau = 0.1$. As shown, the reduced DMRT-QCA phase matrices also exhibit the expected symmetry as found for the Mie case, and the forward scattering also increases as frequency increases. Moreover, the DMRT-QCA phase matrices present more forward scattering than that of the comparable Mie cases. However, we also note that the reduced normalized Mie phase matrix can be steadily and

accurately computed over a wide frequency range (in terms of mean size parameter ka) since there exist numerically stable algorithms [45, 46, 51, 52] for frequencies up to at least ~1000 GHz and for practical hydrometer size distributions. In contrast there is no conclusive study on the stability of the DMRT-QCA algorithm except for a brief discussion of the maximum number N_{max} of L-L equations required for convergence in [37]. From this work N_{max} is suggested to be determined by the relation N_{max}= round($k\langle D\rangle$)+1. This requirement for N_{max} was studied by computing the DMRT-QCA phase matrices at frequencies up to 1000 GHz. First, it should be pointed out that in the three cases of Fig. 2.12 the errors caused by the choice of N_{max} are small (the value of N_{max} is 4 at 100 GHz). As the frequency is extended to 300 GHz a value $N_{max} = 10$ is needed, thus increasing the computational burden. Further, the L-L system of equations becomes ill-conditioned at higher frequencies and a stable numerical solution is currently unavailable. Nonetheless, for microwave remote sensing of snow and ice, DMRT-QCA is still readily computable for the most practical of snow and ice sensing frequencies (i.e., below ~100 GHz).

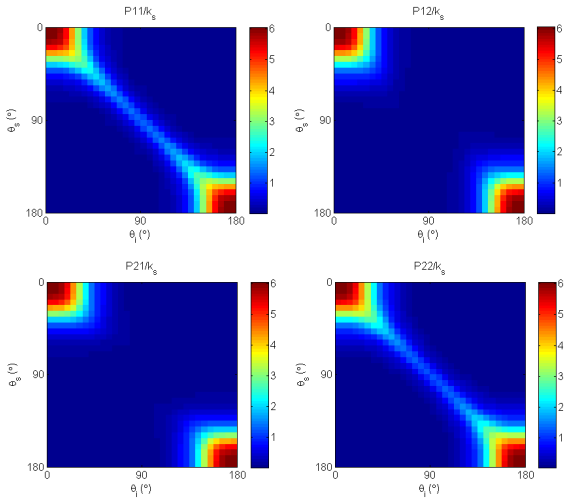


(a) 10 GHz

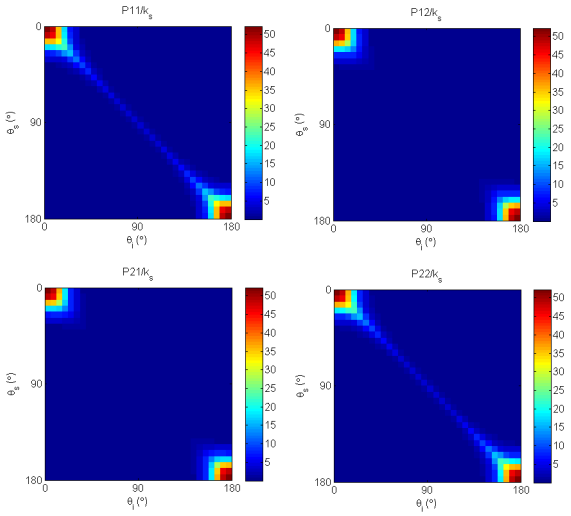
(b) 30 GHz



(c) 100 GHz

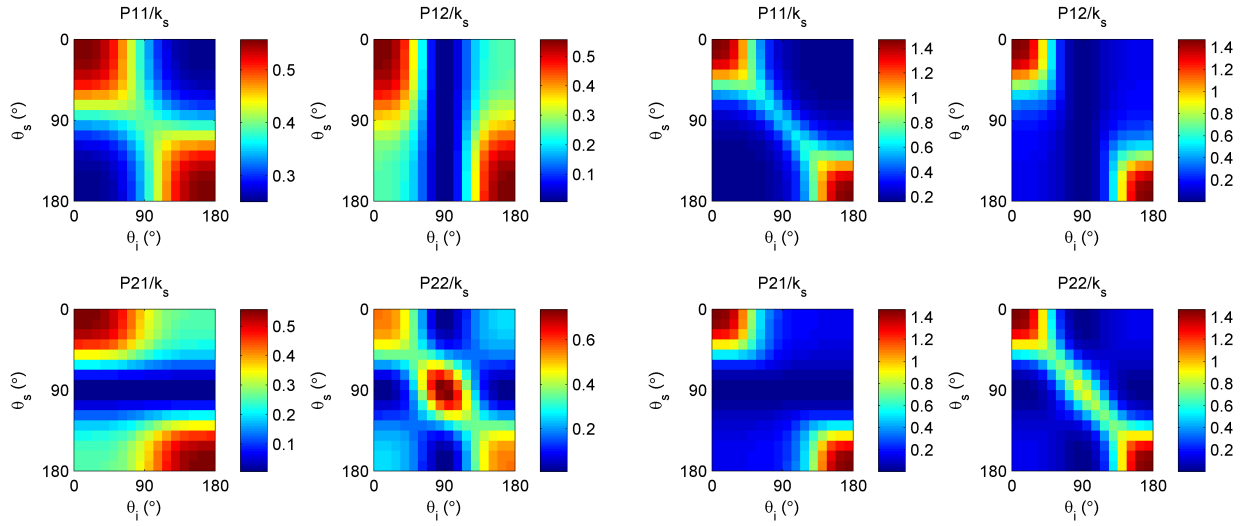


(d) 300 GHz



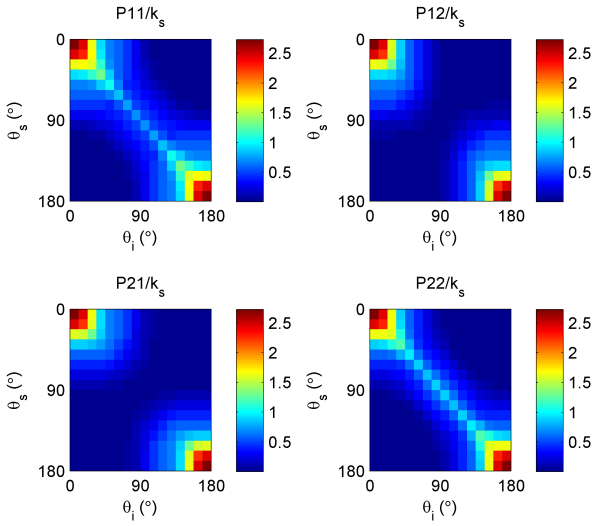
(e) 1000 GHz

Figure 2.11: Reduced normalized Mie phase matrices. Both θ_s and θ_i are discretized into 32 quadrature angles.



(a) 10 GHz

(b) 30 GHz



(c) 100 GHz

Figure 2.12: Reduced normalized DMRT phase matrices. Both θ_s and θ_i are discretized into 16 quadrature angles.

Chapter 3

UMRT Framework for General Planar Stratified Media: Slab Formulation

3.1 DRTE Symmetrization

UMRT assumes a planar stratified medium structure and provides a solution for the brightness temperature $T_B(\theta, z)$ in upwelling $(+)$ and downwelling $(-)$ directions, accounting for polarization coupling caused by the reduced phase matrix. The differential radiative transfer equation (DRTE) is discretized over a set of quadrature angles θ_i, which are determined by the Gauss-Legendre nodes and Christoffel weights:

$$\begin{aligned}\mu_i \tfrac{dT^+_{Bvi}}{dz} = & -k_e T^+_{Bvi} + [\sum_{j=1}^{M} \gamma_j P^{++}_{vvij} T^+_{Bvj} + \sum_{j=1}^{M} \gamma_j P^{+-}_{vvij} T^-_{Bvj} \\ & + \sum_{j=1}^{M} \gamma_j P^{++}_{vhij} T^+_{Bhj} + \sum_{j=1}^{M} \gamma_j P^{+-}_{vhij} T^-_{Bhj}] + k_a T(z)\end{aligned} \tag{3.1}$$

$$\begin{aligned}-\mu_i \tfrac{dT^-_{Bvi}}{dz} = & -k_e T^-_{Bvi} + [\sum_{j=1}^{M} \gamma_j P^{-+}_{vvij} T^+_{Bvj} + \sum_{j=1}^{M} \gamma_j P^{--}_{vvij} T^-_{Bvj} \\ & + \sum_{j=1}^{M} \gamma_j P^{-+}_{vhij} T^+_{Bhj} + \sum_{j=1}^{M} \gamma_j P^{--}_{vhij} T^-_{Bhj}] + k_a T(z)\end{aligned} \tag{3.2}$$

$$\begin{aligned}\mu_i \tfrac{dT^+_{Bhi}}{dz} = & -k_e T^+_{Bhi} + [\sum_{j=1}^{M} \gamma_j P^{++}_{hvij} T^+_{Bvj} + \sum_{j=1}^{M} \gamma_j P^{+-}_{hvij} T^-_{Bvj} \\ & + \sum_{j=1}^{M} \gamma_j P^{++}_{hhij} T^+_{Bhj} + \sum_{j=1}^{M} \gamma_j P^{+-}_{hhij} T^-_{Bhj}] + k_a T(z)\end{aligned} \tag{3.3}$$

$$
\begin{aligned}
-\mu_i \tfrac{dT^-_{Bvi}}{dz} = \; & -k_e T^-_{Bvi} + [\sum_{j=1}^{M} \gamma_j P^{-+}_{hvij} T^+_{Bvj} + \sum_{j=1}^{M} \gamma_j P^{--}_{hvij} T^-_{Bvj} \\
& + \sum_{j=1}^{M} \gamma_j P^{-+}_{hhij} T^+_{Bhj} + \sum_{j=1}^{M} \gamma_j P^{--}_{hhij} T^-_{Bhj}] + k_a T(z)
\end{aligned} \tag{3.4}
$$

where $\mu_i = \cos\theta_i$, γ_j are the Christoffel weights, and M is the number of quadrature angles between zenith and the horizon. Here, we choose $M = 16$ in this study, which is suitable for most passive remote sensing purposes. All μ_i in the above equations are positive as a result of separating the brightness temperature in the up- and down-welling directions. Following [22], and with reference to Eq. (2.31), the discretized reduced phase matrix elements are defined as

$$
\begin{aligned}
P^{++}_{\alpha\beta ij} &= P_{\alpha\beta}(\mu_i, \mu_j) \\
P^{+-}_{\alpha\beta ij} &= P_{\alpha\beta}(\mu_i, -\mu_j) \\
P^{-+}_{\alpha\beta ij} &= P_{\alpha\beta}(-\mu_i, \mu_j) \\
P^{--}_{\alpha\beta ij} &= P_{\alpha\beta}(-\mu_i, -\mu_j)
\end{aligned} \tag{3.5}
$$

where α, β are either v (vertical) or h (horizontal) polarization. As shown in §2 the reduced Mie and DMRT-QCA phase matrices are symmetric with respect to simultaneous permutation of angular indexes and independent permutations of both up- and down-welling indexes:

$$
\begin{aligned}
P^{++}_{\alpha\beta ij} = P^{++}_{\alpha\beta ji}, \quad P^{--}_{\alpha\beta ij} = P^{--}_{\alpha\beta ji} \\
P^{-+}_{\alpha\beta ij} = P^{+-}_{\alpha\beta ji}, \quad P^{++}_{\alpha\beta ij} = P^{--}_{\alpha\beta ij} \\
P^{-+}_{\alpha\beta ij} = P^{+-}_{\alpha\beta ij}
\end{aligned} \tag{3.6}
$$

Following [22], new variables for the up- and down-welling streams are introduced to make the DRTE explicitly symmetric:

$$
\begin{aligned}
u_{vi} = \sqrt{\mu_i \gamma_i} T^+_{Bvi}, \quad u_{hi} = \sqrt{\mu_i \gamma_i} T^+_{Bhi} \\
v_{vi} = \sqrt{\mu_i \gamma_i} T^-_{Bvi}, \quad v_{hi} = \sqrt{\mu_i \gamma_i} T^-_{Bhi}
\end{aligned} \tag{3.7}
$$

Rearranging Eq. (3.1) yields

$$
\begin{aligned}
\tfrac{du_{vi}}{dz} = & \sqrt{\tfrac{\gamma_i}{\mu_i}}\left[-k_{evi}T^{+}_{Bvi}+k_aT\left(z\right)\right] \\
& +\sqrt{\tfrac{\gamma_i}{\mu_i}}\left[\sum_{j=1}^{M}\gamma_jP^{++}_{vvij}T^{+}_{Bvj}+\sum_{j=1}^{M}\gamma_jP^{+-}_{vvij}T^{-}_{Bvj}+\sum_{j=1}^{M}\gamma_jP^{++}_{vhij}T^{+}_{Bhj}+\sum_{j=1}^{M}\gamma_jP^{+-}_{vhij}T^{-}_{Bhj}\right] \\
= & -\sqrt{\tfrac{\gamma_i}{\mu_i}}\left(k_{evi}\delta_{ij}-\sum_{j=1}^{M}\gamma_j\right)T^{+}_{Bvj}+\sqrt{\tfrac{\gamma_i}{\mu_i}}k_aT\left(z\right) \\
& +\sqrt{\tfrac{\gamma_i}{\mu_i}}\left[\sum_{j=1}^{M}\gamma_jP^{+-}_{vvij}T^{-}_{Bvj}+\sum_{j=1}^{M}\gamma_jP^{++}_{vhij}T^{+}_{Bhj}+\sum_{j=1}^{M}\gamma_jP^{+-}_{vhij}T^{-}_{Bhj}\right] \\
= & -\tfrac{1}{\sqrt{\mu_i\gamma_i}}\sqrt{\tfrac{\gamma_i}{\mu_i}}\left(k_{evi}\delta_{ij}-\sum_{j=1}^{M}\gamma_j\right)\underbrace{\left(\sqrt{\mu_i\gamma_i}T^{+}_{Bvj}\right)}_{u_{vi}}+\tfrac{1}{\sqrt{\mu_i\gamma_i}}\sqrt{\tfrac{\gamma_i}{\mu_i}}\sum_{j=1}^{M}\gamma_jP^{+-}_{vvij}\underbrace{\left(\sqrt{\mu_i\gamma_i}T^{-}_{Bvj}\right)}_{v_{vi}} \\
& +\tfrac{1}{\sqrt{\mu_i\gamma_i}}\sqrt{\tfrac{\gamma_i}{\mu_i}}\sum_{j=1}^{M}\gamma_jP^{++}_{vhij}\underbrace{\left(\sqrt{\mu_i\gamma_i}T^{+}_{Bhj}\right)}_{u_{hi}}+\tfrac{1}{\sqrt{\mu_i\gamma_i}}\sqrt{\tfrac{\gamma_i}{\mu_i}}\sum_{j=1}^{M}\gamma_jP^{+-}_{vhij}\underbrace{\left(\sqrt{\mu_i\gamma_i}T^{-}_{Bhj}\right)}_{v_{hi}}+\underbrace{\sqrt{\tfrac{\gamma_i}{\mu_i}}k_aT\left(z\right)}_{f_i} \\
= & -\tfrac{1}{\mu_i}\left(k_{evi}\delta_{ij}-\sum_{j=1}^{M}\gamma_j\right)u_{vi}+\tfrac{1}{\mu_i}\left(\sum_{j=1}^{M}\gamma_jP^{+-}_{vvij}v_{vi}+\sum_{j=1}^{M}\gamma_jP^{++}_{vhij}u_{hi}+\sum_{j=1}^{M}\gamma_jP^{+-}_{vhij}v_{hi}\right)+f_i
\end{aligned}
\tag{3.8}
$$

Similarly rearranging Eqs. (3.2-3.4), the following matrix form of discretized DRTE equations is obtained:

$$
\underbrace{\frac{d}{dz}\begin{bmatrix}\overline{u}_v\\ \overline{u}_h\\ \overline{v}_v\\ \overline{v}_h\end{bmatrix}}_{4M\times 1}=\underbrace{\begin{bmatrix}-\overline{\overline{A}}_0 & -\overline{\overline{C}}_0 & -\overline{\overline{B}}_0 & -\overline{\overline{D}}_0\\ -\overline{\overline{E}}_0 & -\overline{\overline{G}}_0 & -\overline{\overline{F}}_0 & -\overline{\overline{H}}_0\\ \overline{\overline{B}}_0 & \overline{\overline{D}}_0 & \overline{\overline{A}}_0 & \overline{\overline{C}}_0\\ \overline{\overline{F}}_0 & \overline{\overline{H}}_0 & \overline{\overline{E}}_0 & \overline{\overline{G}}_0\end{bmatrix}}_{4M\times 4M}\begin{bmatrix}\overline{u}_v\\ \overline{u}_h\\ \overline{v}_v\\ \overline{v}_h\end{bmatrix}+\underbrace{\begin{bmatrix}\overline{f}\\ \overline{f}\\ -\overline{f}\\ -\overline{f}\end{bmatrix}}_{4M\times 1} \tag{3.9}
$$

$$
\Rightarrow\frac{d}{dz}\begin{bmatrix}\overline{u}\\ \overline{v}\end{bmatrix}=\underbrace{\begin{bmatrix}-\overline{\overline{U}} & -\overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}}_{\text{DOTLRT}}\begin{bmatrix}\overline{u}\\ \overline{v}\end{bmatrix}+\begin{bmatrix}\overline{F}\\ -\overline{F}\end{bmatrix} \tag{3.10}
$$

where $\overline{u}\triangleq\begin{bmatrix}\overline{u}_v\\ \overline{u}_h\end{bmatrix}$, $\overline{v}\triangleq\begin{bmatrix}\overline{v}_v\\ \overline{v}_h\end{bmatrix}$, $\overline{F}\triangleq\begin{bmatrix}\overline{f}\\ \overline{f}\end{bmatrix}$, $\overline{\overline{U}}\triangleq\begin{bmatrix}\overline{\overline{A}}_0 & \overline{\overline{C}}_0\\ \overline{\overline{E}}_0 & \overline{\overline{G}}_0\end{bmatrix}$, and $\overline{\overline{D}}\triangleq\begin{bmatrix}\overline{\overline{B}}_0 & \overline{\overline{D}}_0\\ \overline{\overline{F}}_0 & \overline{\overline{H}}_0\end{bmatrix}$. The sub-matrices for vertical and horizontal polarization are defined as

$$\begin{aligned}
A_{0ij} &= \tfrac{k_e}{\mu_i}\delta_{ij} - \sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{++}_{vvij}, & B_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{+-}_{vvij} \\
C_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{++}_{vhij}, & D_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{+-}_{vhij} \\
E_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{++}_{hvij}, & F_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{+-}_{hvij} \\
G_{0ij} &= \tfrac{k_e}{\mu_i}\delta_{ij} - \sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{++}_{hhij}, & H_{0ij} &= -\sqrt{\tfrac{\gamma_i\gamma_j}{\mu_i\mu_j}}P^{+-}_{hhij}
\end{aligned} \tag{3.11}$$

The vector $\overline{f}$ represents thermal emission from the medium and is defined by $f_i = \sqrt{\frac{\gamma_i}{\mu_i}}k_a T(z)$, where $T(z) \triangleq (T_o - \gamma z)$ and γ is the temperature lapse rate.

Finally, the boundary conditions are

$$\begin{aligned}
\mu_i T^{+}_{\beta i} &= \sum_{j=1}^{M}\gamma_j s_{\beta ij} T^{-}_{B\beta i} + \left(\mu_i - \sum_{j=1}^{M}\gamma_j s_{\beta ji}\right) T_s, \quad z = 0 \\
T^{-}_{\beta i} &= T_{cb}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad z = H
\end{aligned} \tag{3.12}$$

where T_{cb} is the cosmic background temperature at the topmost atmospheric level $z = H$, T_s is the surface background temperature, and in the case of a specular surface the surface bistatic function s_{ij} is obtained using the Fresnel reflection coefficient R_i as

$$s_{\beta ij} = \mu_i \left|R_{\beta i}\right|^2 \delta_{ij}, \quad \gamma_j = 1 \tag{3.13}$$

For coupled vertical-horizontal radiation streams all sub-matrices in $\overline{\overline{U}}$ and $\overline{\overline{D}}$ are defined from a specific reduced phase matrix, which have been shown to be symmetric with respect to incident and scattering angles. Thus both matrices $\overline{\overline{U}}$ and $\overline{\overline{D}}$ are symmetric along with the following two new matrices:

$$\overline{\overline{A}} \triangleq \overline{\overline{U}} + \overline{\overline{D}} = \begin{bmatrix} \overline{\overline{A}}_0 + \overline{\overline{B}}_0 & \overline{\overline{C}}_0 + \overline{\overline{D}}_0 \\ \overline{\overline{E}}_0 + \overline{\overline{F}}_0 & \overline{\overline{G}}_0 + \overline{\overline{H}}_0 \end{bmatrix} \tag{3.14}$$

and

$$\overline{\overline{B}} \triangleq \overline{\overline{U}} - \overline{\overline{D}} = \begin{bmatrix} \overline{\overline{A}}_0 - \overline{\overline{B}}_0 & \overline{\overline{C}}_0 - \overline{\overline{D}}_0 \\ \overline{\overline{E}}_0 - \overline{\overline{F}}_0 & \overline{\overline{G}}_0 - \overline{\overline{H}}_0 \end{bmatrix} \tag{3.15}$$

In order to apply the stable matrix inversion technique of DOTLRT, proof of the matrix $\overline{\overline{U}}+\overline{\overline{D}}$ being positive definite is as follows:

(1) Separate the matrix $\overline{\overline{U}}+\overline{\overline{D}}$ into the sum of two matrices: one is a diagonal matrix, denoted as $\overline{\overline{S}}_{\rm d}$ and the other is defined by $\overline{\overline{U}}+\overline{\overline{D}}-\overline{\overline{S}}_{\rm d}$, denoted as $\overline{\overline{S}}_{\rm r}$. The problem is now to prove the positive definiteness of both of these matrices. By design the matrix $\overline{\overline{S}}_{\rm d}$ has elements:

$$\overline{\overline{S}}_{\rm d}=\left[\begin{array}{cc}\overline{\overline{S}}_{{\rm d}v} & \overline{\overline{0}}\\ \overline{\overline{0}} & \overline{\overline{S}}_{{\rm d}h}\end{array}\right] \tag{3.16}$$

$$\left\{\overline{\overline{S}}_{{\rm d}\beta}\right\}_{ij}=\frac{k_{e\beta i}\delta_{ij}}{\mu_i}-\frac{\delta_{ij}}{\mu_i}\sum_{k=1}^{M}\gamma_k\left(P^{++}_{v\beta ki}+P^{+-}_{v\beta ki}++P^{++}_{h\beta ki}+P^{+-}_{h\beta ki}\right)\ ,\quad \beta=v \text{ or } h \tag{3.17}$$

Note that Eq. (3.17) is indeed the discretized form of $k_{e\beta}-k_{s\beta}$, which is always positive for passive media due to inevitable small losses.

(2) From the above definition, the matrix $\overline{\overline{S}}_{\rm r}$ is defined as

$$\overline{\overline{S}}_{\rm r}=\left[\begin{array}{cc}\underbrace{\overline{\overline{S}}_{{\rm r}v}}_{2M\times M} & \underbrace{\overline{\overline{S}}_{{\rm r}h}}_{2M\times M}\end{array}\right] \tag{3.18}$$

$$\left\{\overline{\overline{S}}_{{\rm r}\beta}\right\}_{ij}=\begin{array}{l}\frac{1}{\mu_i}\sum_{k=1}^{M}\gamma_k\left(P^{++}_{v\beta ki}+P^{+-}_{v\beta ki}+P^{++}_{h\beta ki}+P^{+-}_{h\beta ki}\right)\delta_{ij}\\ -\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(P^{++}_{v\beta ij}+P^{+-}_{v\beta ij}+P^{++}_{h\beta ij}+P^{+-}_{h\beta ij}\right)\end{array}\ ,\quad \beta=v \text{ or } h \tag{3.19}$$

Now consider the eigenvalues and eigenvectors of $\overline{\overline{S}}_{\rm r}$, where $\overline{\overline{S}}_{\rm r}\overline{u}=\lambda\overline{u}$. Following the development of Gershgorin's circle theorem the maximal value of the ratio $u_i/\sqrt{\mu_i\gamma_i}$ for all i is found

$$\left|\frac{u_{i_0}}{\sqrt{\mu_{i_0}\gamma_{i_0}}}\right|>\frac{u_i}{\sqrt{\mu_i\gamma_i}} \tag{3.20}$$

Assuming that $u_{i_0} > 0$ and noting that the phase matrix is symmetric, it follows that

$$\begin{aligned}
\lambda u_{i_0} = & \frac{u_{i_0}}{\mu_{i_0}} \sum_{k=1}^{M} \gamma_k \left(P^{++}_{v\beta ki} + P^{+-}_{v\beta ki} + P^{++}_{h\beta ki} + P^{+-}_{h\beta ki} \right) \\
& - \sqrt{\frac{\gamma_{i_0}}{\mu_{i_0}}} \sum_{j=1}^{M} \frac{u_j \gamma_j}{\sqrt{\mu_j \gamma_j}} \left(P^{++}_{v\beta ij} + P^{+-}_{v\beta ij} + P^{++}_{h\beta ij} + P^{+-}_{h\beta ij} \right) \\
\geq & \frac{u_{i_0}}{\mu_{i_0}} \sum_{k=1}^{M} \gamma_k \left(P^{++}_{v\beta ki} + P^{+-}_{v\beta ki} + P^{++}_{h\beta ki} + P^{+-}_{h\beta ki} \right) \qquad , \quad \beta = v \text{ or } h \qquad (3.21) \\
& - \sqrt{\frac{\gamma_{i_0}}{\mu_{i_0}}} \frac{u_{i_0}}{\sqrt{\mu_{i_0} \gamma_{i_0}}} \sum_{j=1}^{M} \gamma_j \left(P^{++}_{v\beta ij} + P^{+-}_{v\beta ij} + P^{++}_{h\beta ij} + P^{+-}_{h\beta ij} \right) \\
= & \ 0
\end{aligned}$$

(3) A similar argument can be applied to the case of the matrix $\overline{\overline{U}} - \overline{\overline{D}}$. Hence, we conclude that both matrices $\overline{\overline{U}} + \overline{\overline{D}}$ and $\overline{\overline{U}} - \overline{\overline{D}}$ are symmetric and positive definite, and therefore applicable to the stable inversion technique used within DOTLRT.

3.2 Solution for a Single Layer

UMRT assumes a planar-stratified stack of reciprocal homogenous layers in which the medium properties are assumed constant and the source vector $\overline{F}$ is at most linear in height (Fig. 3.1(a)). In addition to the extension to multiple coupled Stokes parameters, the assumption of the kinetic temperature of a layer being linear is another fundamental difference between UMRT and DOTLRT.

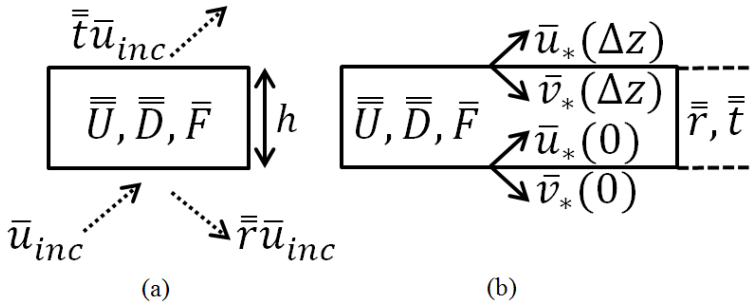


Figure 3.1: (a) Matrix representation of the reflection and transmission matrix operators, and (b) self-radiation stream vectors for a single layer.

Owing to thermal emission, the layer will generate self-radiation streams in the up- (u) and

down- (v) welling directions at its top and bottom surfaces, respectively. Such streams denoted by the subscript $*$ (Fig. 3.1(b)). To solve for them it is required to compute the reflection and transmission matrices ($\overline{\overline{r}}$ and $\overline{\overline{t}}$) which describe the volumetric scattering inside the layer. The layer is assumed to be embedded within a homogeneous dielectric environment of permittivity equal to the effective permittivity of the layer. As such, there is no surface Fresnel reflection at the interfaces to this neutral dielectric background environment. Assuming an external radiation field $\overline{u}_{inc}$ incident from the bottom of the layer (Fig. 3.1(a)), we define the reflection and transmission matrices implicitly by

$$\begin{aligned} \overline{u} &= \overline{\overline{t}}\,\overline{u}_{inc}, \quad \text{at } z = h \\ \overline{v} &= \overline{\overline{r}}\,\overline{u}_{inc}, \quad \text{at } z = 0 \end{aligned} \tag{3.22}$$

These matrices can be found using the homogeneous solution to the DRTE:

$$\frac{d}{dz}\begin{bmatrix} \overline{u}\,(z) \\ \overline{v}\,(z) \end{bmatrix} = \begin{bmatrix} -\overline{\overline{U}} & -\overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix} \begin{bmatrix} \overline{u}\,(z) \\ \overline{v}\,(z) \end{bmatrix} \tag{3.23}$$

, given as

$$\begin{bmatrix} \overline{u}\,(z) \\ \overline{v}\,(z) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \overline{\overline{c}} & -\overline{\overline{s}}\,\overline{\overline{A}} \\ -\overline{\overline{B}}\,\overline{\overline{s}} & \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}\,(0) \\ \overline{v}\,(0) \end{bmatrix} \tag{3.24}$$

The details of the above form of the DRTE homogeneous solution are provided in Appendix C.

Applying Eq. (3.22) we have

$$\begin{bmatrix} \overline{\overline{t}}\,\overline{u}_{inc} \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \overline{\overline{c}} - \overline{\overline{B}}\,\overline{\overline{s}} - \overline{\overline{s}}\,\overline{\overline{A}} + \overline{\overline{c}}^{T} & -\overline{\overline{c}} + \overline{\overline{B}}\,\overline{\overline{s}} - \overline{\overline{s}}\,\overline{\overline{A}} + \overline{\overline{c}}^{T} \\ -\overline{\overline{c}} - \overline{\overline{B}}\,\overline{\overline{s}} + \overline{\overline{s}}\,\overline{\overline{A}} + \overline{\overline{c}}^{T} & \overline{\overline{c}} + \overline{\overline{B}}\,\overline{\overline{s}} + \overline{\overline{s}}\,\overline{\overline{A}} + \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} \overline{u}_{inc} \\ \overline{\overline{r}}\,\overline{u}_{inc} \end{bmatrix} \tag{3.25}$$

where the matrices $\overline{\overline{c}}$ and $\overline{\overline{s}}$ are defined as follows

$$\begin{aligned} \overline{\overline{c}} &\triangleq \cosh\left(\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}h\right) \\ \overline{\overline{s}} &\triangleq \sinh\left(\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}h\right) \cdot \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} \end{aligned} \tag{3.26}$$

and are calculated at $z = h$. From Eq. (3.25) we obtain

$$\begin{aligned} \bar{\bar{t}} = &\quad 2\left(\bar{\bar{c}} + \bar{\bar{B}}\,\bar{\bar{s}} + \bar{\bar{s}}\,\bar{\bar{A}} + \bar{\bar{c}}^{T}\right)^{-1} \triangleq 2\bar{\bar{Q}}^{-1} \\ \bar{\bar{r}} = &\quad \bar{\bar{Q}}^{-1}\left(\bar{\bar{c}} + \bar{\bar{B}}\,\bar{\bar{s}} - \bar{\bar{s}}\,\bar{\bar{A}} - \bar{\bar{c}}^{T}\right) \end{aligned} \tag{3.27}$$

It is noted that the above matrices exhibit the symmetry necessary as a result of reciprocity

$$\bar{\bar{r}} = \bar{\bar{r}}^{T} \text{ and } \bar{\bar{t}} = \bar{\bar{t}}^{T} \tag{3.28}$$

Although the above procedure is analytically correct, the direct inversion of the matrix $\bar{\bar{Q}}$ will usually fail numerically when the medium layer is either highly opaque or thick or both. The problem is that the matrix functions $\bar{\bar{c}}$ and $\bar{\bar{s}}$ are functions of $\cosh(x)$ and $\sinh(x)$, respectively. Such hyperbolic functions contain fast growing exponentials which quickly lead to ill-conditioning of $\bar{\bar{Q}}$. To circumvent this problem DOTLRT uses following aforementioned equations, Eqs. (1.2-1.6) in Chapter 1:

$$\begin{aligned} \bar{\bar{A}} &= \bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1 \bar{\bar{M}}_1^{T} \\ \bar{\bar{M}}_1 \bar{\bar{M}}_1^{T} &= \bar{\bar{M}}_1 \bar{\bar{M}}_1^{-1} = \bar{\bar{I}} \\ \bar{\bar{\Lambda}}_1^{\frac{1}{2}} \bar{\bar{M}}_1^{T} \bar{\bar{B}}\, \bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1^{\frac{1}{2}} &= \bar{\bar{M}}_2 \bar{\bar{\Lambda}}_2 \bar{\bar{M}}_2^{T} \\ \bar{\bar{A}}\,\bar{\bar{B}} &= \left(\bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1^{\frac{1}{2}} \bar{\bar{M}}_2\right) \bar{\bar{\Lambda}}_2 \left(\bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1^{-\frac{1}{2}} \bar{\bar{M}}_2\right)^{T} \\ g\left(\bar{\bar{A}}\,\bar{\bar{B}}\right) &= \left(\bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1^{\frac{1}{2}} \bar{\bar{M}}_2\right) g\left(\bar{\bar{\Lambda}}_2\right) \left(\bar{\bar{M}}_1 \bar{\bar{\Lambda}}_1^{-\frac{1}{2}} \bar{\bar{M}}_2\right)^{T} \end{aligned} \tag{3.29}$$

for any arbitrary analytical function g operated on the constituent symmetric and positive definite matrices $\bar{\bar{A}}$ and $\bar{\bar{B}}$ along with analytical diagonalization and factorization of the both matrices that comprise $\bar{\bar{Q}}$ to represent it as

$$\bar{\bar{Q}} = \bar{\bar{M}}_1 \bar{\bar{a}}\, \bar{\bar{\zeta}}\, \bar{\bar{b}}_t \bar{\bar{M}}_1^{T} \tag{3.30}$$

Thus, the matrices $\bar{\bar{r}}$ and $\bar{\bar{t}}$ are readily computed as follows:

$$\begin{aligned} \overline{\overline{t}} &= 2\overline{\overline{M}}_1 \overline{\overline{b}}_t^{-1} \overline{\overline{\zeta}}^{-1} \overline{\overline{a}}^{-1} \overline{\overline{M}}_1^T \\ \overline{\overline{r}} &= \overline{\overline{t}} - \overline{\overline{M}}_1 \overline{\overline{b}}_t^{-1} \overline{\overline{b}}_r \overline{\overline{M}}_1^T \end{aligned} \tag{3.31}$$

where as defined in Eqs. (51-56) in [22]:

$$\begin{aligned} \overline{\overline{\zeta}} &= \overline{\overline{\Lambda}}_2^{-\frac{1}{2}} \sinh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}} h\right) \\ \overline{\overline{a}} &= \overline{\overline{\Lambda}}_1^{\frac{1}{2}} \overline{\overline{M}}_2 + \overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \overline{\overline{M}}_2 \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \coth\left(\tfrac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}} h\right) \\ \overline{\overline{b}}_t &= \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{\frac{1}{2}} + \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \tanh\left(\tfrac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}} h\right) \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \\ \overline{\overline{b}}_r &= \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{\frac{1}{2}} - \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \tanh\left(\tfrac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}} h\right) \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \end{aligned}$$

In Eqs. (3.30-3.31), the matrix $\overline{\overline{M}}_1$ is the orthogonal matrix consisting of the column eigenvectors of the matrix $\overline{\overline{A}}$. The matrices $\overline{\overline{b}}_t$, $\overline{\overline{b}}_r$ and $\overline{\overline{a}}$ are transitional matrix functions involving $\tanh(x)$ and $\coth(x)$, where x is a function of the layer thickness. Since both $\tanh(x)$ and $\coth(x)$ are bounded to 1 as $x \to \infty$ the matrices $\overline{\overline{b}}_t$, $\overline{\overline{b}}_r$ and $\overline{\overline{a}}$ tend to finite limits. The matrix $\overline{\overline{\zeta}}$ is a diagonal matrix function containing terms in $\sinh(x)$. Since $\overline{\overline{\zeta}}$ is diagonal, it is always precisely invertible.

For a layer with constant temperature profile (i.e., $\gamma = 0$) DOTLRT computes the self-radiation stream vectors as follows

$$\begin{aligned} \overline{u}_*(0) = \overline{v}_*(0) &= \left(\overline{\overline{I}} - \overline{\overline{r}} - \overline{\overline{t}}\right) \overline{u}_{inh} \\ \overline{u}_*(0) &= \overline{u}_*(h) \\ \overline{v}_*(0) &= \overline{v}_*(h) \end{aligned} \tag{3.32}$$

where $\overline{\overline{I}} - \overline{\overline{r}} - \overline{\overline{t}}$ can be interpreted as an effective emissivity matrix for the layer. The inhomogeneous solution of the DRTE

$$\frac{d}{dz}\begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} = \begin{bmatrix} -\overline{\overline{U}} & -\overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} + \begin{bmatrix} \overline{F} \\ -\overline{F} \end{bmatrix} \tag{3.33}$$

is given by

$$\frac{d}{dz}\begin{bmatrix}\overline{u}(z)\\ \overline{v}(z)\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}=\begin{bmatrix}-\overline{\overline{U}} & -\overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}\begin{bmatrix}\overline{u}\\ \overline{v}\end{bmatrix}+\begin{bmatrix}\overline{F}\\ -\overline{F}\end{bmatrix} \tag{3.34}$$

Arranging Eq. (3.34) gives

$$\begin{bmatrix}\overline{\overline{U}} & \overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}\begin{bmatrix}\overline{u}(z)\\ \overline{v}(z)\end{bmatrix}=\begin{bmatrix}\overline{F}\\ \overline{F}\end{bmatrix} \tag{3.35}$$

$$\Rightarrow\left(\overline{\overline{U}}+\overline{\overline{D}}\right)\overline{u}(z)=\left(\overline{\overline{U}}+\overline{\overline{D}}\right)\overline{v}(z) \tag{3.36}$$

Thus, $\overline{u}(z)=\overline{v}(z)$. Substituting $\overline{u}(z)=\overline{v}(z)$ into Eq. (3.35) yields following inhomogeneous solution:

$$\begin{bmatrix}\overline{u}_{inh}(z)\\ \overline{v}_{inh}(z)\end{bmatrix}=\begin{bmatrix}\left(\overline{\overline{U}}+\overline{\overline{D}}\right)^{-1}\overline{F}\\ \left(\overline{\overline{U}}+\overline{\overline{D}}\right)^{-1}\overline{F}\end{bmatrix}=\begin{bmatrix}\overline{\overline{A}}^{-1}\overline{F}\\ \overline{\overline{A}}^{-1}\overline{F}\end{bmatrix} \tag{3.37}$$

Extending the above to the case of a linear temperature profile the inhomogeneous DRTE is solved by assuming $\overline{u}_{inh}(z)=\overline{u}_o-\overline{u}_1 z$, $\overline{v}_{inh}(z)=\overline{v}_o-\overline{v}_1 z$, $F_i(z)\triangleq\sqrt{\frac{\gamma_i}{\mu_i}}k_a(T_o-\gamma z)\triangleq F_{oi}-\gamma_{T_i}z$, where $F_{oi}=\sqrt{\frac{\gamma_i}{\mu_i}}k_aT_o$ and $\gamma_{T_i}=\sqrt{\frac{\gamma_i}{\mu_i}}k_a\gamma$. Substituting these quantities into the DRTE yields

$$\begin{bmatrix}-\overline{u}_1\\ -\overline{v}_1\end{bmatrix}=\begin{bmatrix}-\overline{\overline{U}} & -\overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}\begin{bmatrix}\overline{u}_o-\overline{u}_1 z\\ \overline{v}_o-\overline{v}_1 z\end{bmatrix}+\begin{bmatrix}\overline{F}_o-\overline{\gamma}_T z\\ -\overline{F}_o+\overline{\gamma}_T z\end{bmatrix} \tag{3.38}$$

Balancing Eq. (3.38), the terms with z dependence vanish, leading to

$$\begin{bmatrix}-\overline{\overline{U}} & -\overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}\begin{bmatrix}\overline{u}_1\\ \overline{v}_1\end{bmatrix}=\begin{bmatrix}-\overline{\gamma}_T\\ \overline{\gamma}_T\end{bmatrix}$$

$$\Rightarrow\quad\begin{bmatrix}\overline{u}_1\\ \overline{v}_1\end{bmatrix}=\begin{bmatrix}\overline{\overline{U}} & \overline{\overline{D}}\\ \overline{\overline{D}} & \overline{\overline{U}}\end{bmatrix}^{-1}\begin{bmatrix}\overline{\gamma}_T\\ \overline{\gamma}_T\end{bmatrix} \tag{3.39}$$

To explicitly solve Eq. (3.39) we apply block matrix inversion [53] and obtain

$$\begin{bmatrix} \overline{\overline{U}} & \overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} & -\overline{\overline{U}}^{-1}\overline{\overline{D}}\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} \\ -\overline{\overline{U}}^{-1}\overline{\overline{D}}\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} & \left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} \end{bmatrix} \tag{3.40}$$

Applying Eq. (3.40) to Eq. (3.39) we have

$$\begin{bmatrix} \overline{u}_1 \\ \overline{v}_1 \end{bmatrix} = \begin{bmatrix} \left(\overline{\overline{I}} - \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1}\overline{\gamma}_T \\ \left(\overline{\overline{I}} - \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1}\overline{\gamma}_T \end{bmatrix} \tag{3.41}$$

The above solution can be simplified using the following derivation:

$$\begin{aligned} & \left(\overline{\overline{U}} + \overline{\overline{D}}\right)\left[\left(\overline{\overline{I}} - \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1}\right] \\ = \; & \left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} = \overline{\overline{I}} \end{aligned} \tag{3.42}$$

$$\Rightarrow \left(\overline{\overline{I}} - \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} = \left(\overline{\overline{U}} + \overline{\overline{D}}\right)^{-1} = \overline{\overline{A}}^{-1} \tag{3.43}$$

From Eq. (3.43), Eq. (3.41) is equivalent to the following:

$$\begin{bmatrix} \overline{u}_1 \\ \overline{v}_1 \end{bmatrix} = \begin{bmatrix} \overline{\overline{A}}^{-1}\overline{\gamma}_T \\ \overline{\overline{A}}^{-1}\overline{\gamma}_T \end{bmatrix} \tag{3.44}$$

Since $\overline{u}_1 = \overline{v}_1$ the remainder of Eq. (3.38) yields

$$\begin{aligned} \begin{bmatrix} \overline{u}_o \\ \overline{v}_o \end{bmatrix} &= \begin{bmatrix} \overline{\overline{U}} & \overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix}^{-1} \begin{bmatrix} \overline{F}_o + \overline{u}_1 \\ \overline{F}_o - \overline{v}_1 \end{bmatrix} \\ &= \begin{bmatrix} \overline{\overline{A}}^{-1}\overline{F}_o + \left(\overline{\overline{I}} + \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1}\overline{u}_1 \\ \overline{\overline{A}}^{-1}\overline{F}_o - \left(\overline{\overline{I}} + \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1}\overline{v}_1 \end{bmatrix} \end{aligned} \tag{3.45}$$

Similarly, it can be shown that

$$\left(\overline{\overline{I}} + \overline{\overline{U}}^{-1}\overline{\overline{D}}\right)\left(\overline{\overline{U}} - \overline{\overline{D}}\,\overline{\overline{U}}^{-1}\overline{\overline{D}}\right)^{-1} = \left(\overline{\overline{U}} - \overline{\overline{D}}\right)^{-1} = \overline{\overline{B}}^{-1} \tag{3.46}$$

Thus,

$$\begin{bmatrix} \overline{u}_o \\ \overline{v}_o \end{bmatrix} = \begin{bmatrix} \overline{\overline{A}}^{-1}\overline{F}_o + \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \\ \overline{\overline{A}}^{-1}\overline{F}_o - \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \end{bmatrix} \tag{3.47}$$

Finally, from Eqs. (3.44) and (3.47), the inhomogeneous solution of Eq. (3.33) is

$$\begin{aligned} \begin{bmatrix} \overline{u}_{inh}(z) \\ \overline{v}_{inh}(z) \end{bmatrix} &= \begin{bmatrix} \overline{\overline{A}}^{-1}\overline{F}_o + \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T - \overline{\overline{A}}^{-1}\overline{\gamma}_T z \\ \overline{\overline{A}}^{-1}\overline{F}_o - \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T - \overline{\overline{A}}^{-1}\overline{\gamma}_T z \end{bmatrix} \\ &= \begin{bmatrix} \overline{\overline{A}}^{-1}\left(\overline{F}_o - \overline{\gamma}_T z\right) + \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \\ \overline{\overline{A}}^{-1}\left(\overline{F}_o - \overline{\gamma}_T z\right) - \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \end{bmatrix} \\ &= \begin{bmatrix} \overline{\overline{A}}^{-1}\overline{F}(z) + \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \\ \overline{\overline{A}}^{-1}\overline{F}(z) - \overline{\overline{B}}^{-1}\overline{\overline{A}}^{-1}\overline{\gamma}_T \end{bmatrix} \end{aligned} \tag{3.48}$$

Note that if $\gamma = 0$ Eq. (3.48) is reduced to Eq. (3.37). The connection between the inhomogeneous solutions, $\overline{u}_{inh}$ and $\overline{v}_{inh}$, and the upwelling self-radiation stream vector $\overline{u}_*$ is illustrated in Fig. 3.2 where two artificial external radiation stream vectors $\overline{v}_{inh}(h)$ incident on the top of the layer from above and $\overline{u}_{inh}(0)$ incident on the bottom of the layer from below are assumed. The two incident stream vectors will correspondingly produce two additional stream vectors at the top of the layer equal to $\overline{\overline{r}}\,\overline{v}_{inh}(h)$ and $\overline{\overline{t}}\,\overline{u}_{inh}(0)$, illustrated in Fig. 3.2(b).

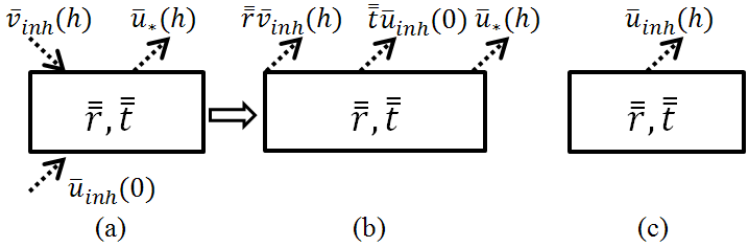


Figure 3.2: Calculation of the upwelling self-radiation for a single layer with linear temperature profile.

Adding all of the stream components in Fig. 3.2(b) results in the following expression for the upwelling inhomogeneous solution at the top of the layer in terms of $\overline{\overline{r}}$ and $\overline{\overline{t}}$:

$$\overline{u}_{inh}(h) = \overline{u}_{*}(h) + \overline{\overline{r}}\,\overline{v}_{inh}(h) + \overline{\overline{t}}\overline{u}_{inh}(0) \tag{3.49}$$

Rearranging Eq. (3.49) the upwelling self-radiation stream vector can be expressed as

$$\overline{u}_{*}(h) = \overline{u}_{inh}(h) - \overline{\overline{r}}\,\overline{v}_{inh}(h) - \overline{\overline{t}}\overline{u}_{inh}(0) \tag{3.50}$$

Similarly, the downwelling self-radiated stream vector is

$$\overline{v}_{*}(0) = \overline{v}_{inh}(0) - \overline{\overline{r}}\,\overline{u}_{inh}(0) - \overline{\overline{t}}\overline{v}_{inh}(h) \tag{3.51}$$

The above solutions (3.50) and (3.51) extend DOTLRT to make UMRT a more widely applicable polarimetric (three Stokes parameter) and level-centric (rather than layer-centric) discrete-ordinate radiative transfer solution .

3.3 Solution for a Multilayer Stack

Using the single-layer solution a procedure for solving for the total radiated and reflected stream vectors for a multilayer stack with non-refracting boundaries can be developed. Once the matrices $\overline{\overline{r}}$ and $\overline{\overline{t}}$ for all individual layers and the vectors $\overline{u}_{*}$ and $\overline{v}_{*}$ at all levels are obtained, the overall radiative characteristics of the stack, $\overline{\overline{R}}^{(n+1)}$ and $\overline{U}_{*}^{(n+1)}$ can be calculated by upward recursion. Since the vectors $\overline{u}_{*}$ and $\overline{v}_{*}$ are fundamentally different for a layer with linear temperature profile versus a constant-temperature layer the upward recursive formulae of DOTLRT are modified as follows

$$\begin{aligned} \overline{U}_{*}^{(n+1)} = \; & \overline{u}_{*}^{(n+1)} + \overline{\overline{t}}^{(n+1)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)} \overline{\overline{r}}^{(n+1)} \right)^{-1} \\ & \cdot \left(\overline{U}_{*}^{(n)} + \overline{\overline{R}}^{(n)} \overline{v}_{*}^{(n+1)} \right) \end{aligned} \tag{3.52}$$

$$\overline{\overline{R}}^{(n+1)} = \overline{\overline{r}}^{(n+1)} + \overline{\overline{t}}^{(n+1)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)} \overline{\overline{r}}^{(n+1)} \right)^{-1} \overline{\overline{R}}^{(n)} \overline{\overline{t}}^{(n+1)} \tag{3.53}$$

where the uppercase characters denote characteristics of a stack in order to distinguish them with their counterparts for a single layer.

The boundary conditions for the stack are:

$$\overline{U}_*^{(0)} = \overline{F}^{(0)} \text{ and } \overline{\overline{R}}^{(0)} = \overline{\overline{S}} \tag{3.54}$$

where $\overline{F}^{(0)}$ denotes the upwelling stream vector from the bounding lower half space and $\overline{\overline{S}}$ is defined by the surface bistatic scattering function of the lower half space

$$S_{\beta ij} = \sqrt{\frac{\gamma_i \gamma_j}{\mu_i \mu_j}} s_{\beta ij} \tag{3.55}$$

It should be noted point that all of the above stream vectors ($\overline{U}_*$, $\overline{u}_*$, $\overline{v}_*$), reflection matrices ($\overline{\overline{R}}$, $\overline{\overline{r}}$) and transmission matrices ($\overline{\overline{t}}$) are calculated assuming that each single layer or stack has non-refractive boundaries. For this simple case the UMRT Jacobian procedure is analogous to that of DOTLRT except for two aspects: 1) the phase matrix is extended in polarization, including the exact Mie and DMRT-QCA phase matrices so that the associated Jacobian calculations must correspondingly be extended, and 2) the physical temperature profile of a layer is extended from being constant to linear, thus the UMRT Jacobian includes the temperature lapse rate and the difference in the up- and down-welling self-radiation streams needs to be considered. Further, if the stack contains refractive boundaries then the above recursive equations need to be correspondingly modified, along with the UMRT Jacobian procedure. The details of such a refractive extension are provided in Chapter 4.

3.4 Numerical Examples

The calculations of self-radiation streams and reflection and transmission matrices for a single layer are validated by imposing energy conservation. The validation scheme is depicted in Fig. 5.2. In Fig. 5.2, we assume a single layer with constant physical temperature T_o is embedded in a homogeneous background environment whose physical temperature is also T_o. This scenario

Table 3.1: Validation of four phase matrices using energy conservation.

Test Conditions	Max. Error (K)
HG, Water, Frequencies up to 10^3 GHz	$\sim 10^{-13}$
Rayleigh, Water, Frequencies up to 10^3 GHz	$\sim 10^{-13}$
Mie, Water, Frequencies up to 10^3 GHz	$\sim 10^{-13}$
DMRT, Dry Snow, Frequencies up to 10^2 GHz	$\sim 10^{-10}$

results in down- and up-welling radiation streams $\overline{v}_{inc}$ and $\overline{u}_{inc}$ impinging on the layer. From thermodynamic equilibrium the brightness temperature of the sum of $\overline{u}_* + \overline{\overline{r}}\,\overline{v}_{inc} + \overline{\overline{t}}\overline{u}_{inc}$ must equal to T_o at all observation angles. As recorded in Tab. 3.1, the UMRT model was tested for this condition using four reduced phase matrices (HG, Rayleigh, Mie, and DMRT-QCA), and two nominal materials (water and dry snow). For the first three reduced phase matrices, the validation was performed up to 1000 GHz in frequency, employing a single rain layer model with 1 km thickness and under the Marshall-Palmer (MP) size distribution with precipitation = 10 mm/hr. For the reduced DMRT-QCA phase matrix, validation was performed at frequencies up to 100 GHz using a 1 m thick dry snow layer with the following parameters: $f_v = 0.25$, $\tau = 0.1$, and $\langle D \rangle = 1.4$ mm. The error between the brightness temperature computed by UMRT and T_o at the i^{th} angle is defined as

$$\varepsilon_{\max}(\theta_i) = \left| \frac{\left(\overline{u}_* + \overline{\overline{r}}\overline{v}_{inc} + \overline{\overline{t}}\overline{u}_{inc}\right)_i}{\sqrt{\mu_i \gamma_i}} - T_o \right| \tag{3.56}$$

As shown in Tab. 3.1, the maximum absolute error of the various phase matrix cases is of order 10^{-10} to 10^{-13} K over sixteen discrete observation angles. This error can be ascribed to roundoff error associated with **IEEE** standard arithmetic.

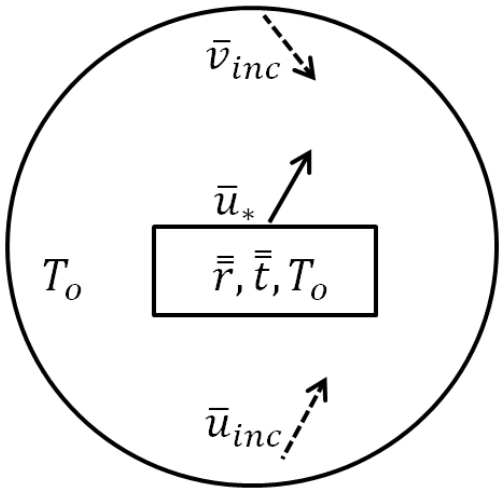


Figure 3.3: Validation of the UMRT single layer solution by imposing energy conservation

Next, a single rain layer is used as an example to compare the brightness temperatures computed by UMRT assuming each of three reduced phase matrices: Rayleigh, HG, and Mie. Both constant and linear temperature profiles were tested for this layer. The MP size distribution is assumed and the water permittivity is determined by Meissner and Wentz's double Debye expression [50]. The test details are provided in Tab. 3.2.

Table 3.2: Single rain layer under the Marshall-Palmer (MP) size distribution.

Layer	Temperature (K)	Rain Rate	$\langle D \rangle$ (mm)	Frequency (GHz)
Water, 1 km	300	10 mm/hr	0.40	10.7, 18.6, 37.0
Water, 1 km	300 to 273	10 mm/hr	0.40	10.7, 18.6, 37.0

For the layer with a constant 300 K temperature the emitted brightness temperatures were computed for the three phase matrices at three distinct frequencies (10.7, 18.6 and 37.0 GHz), and categorized by direction (up- and down-welling) and polarization (horizontal and vertical) (see Fig. 3.4). From Fig. 3.4, we see that at the two low frequencies (10.7 and 18.6 GHz) the brightness temperatures for the three phase matrices are nearly identical while at the high frequency (37.0 GHz), the Rayleigh and HG cases are slightly colder (~2 K) than that of the Mie case.

We also note that all three phase matrices yield identical upwelling and downwelling radiation streams, which is expected for a uniform temperature profile. We also see that in both Rayleigh and HG, the horizontal brightness temperatures equal their vertical counterparts, as expected from the decoupled polarization characteristic of both Rayleigh and HG phase matrices. However, the Mie phase matrix shows clear differences between the horizontal and vertical brightness temperatures of ~6 K, particularly at the frequencies approaching the transition from the Rayleigh to the Mie region.

For the layer with a linear temperature profile the brightness temperatures are plotted in Fig. 3.5. Comparing Figs. 3.4 and 3.5 there is similar general behavior in both cases. However, the downwelling radiation streams are considerably greater than their corresponding upwelling counterparts, which is what is expected given the linear temperature profile.

Similarly, the polarized brightness temperatures emitted in the up- and down-welling directions from a single dry snow layer with 0.1 m thickness at four frequencies (10.7, 18.6, 37.0 and 89.0 GHz) were studied (Figs. 3.6 and 3.7). The test details are provided in Tab. 3.3. From Figs. 3.6 and 3.7, the differences in brightness temperatures due to different temperature profiles are seen. It is noted that the difference between vertical and horizontal brightness temperatures is much significant at the high frequency (>~10 K, 89.0 GHz) than that at other lower frequencies. It is also noted that as frequency increases, the brightness temperatures at normal incidence decrease, which is what expected since the snow layer appears less emissive at higher frequencies.

Table 3.3: Single dry snow layer using the reduced DMRT-QCA phase matrix.

Layer	Temperature (K)	f_v	τ	$\langle D \rangle$ (mm)	Frequency (GHz)
Snow, 0.1 m	273	0.25	0.1	1.4	10.7, 18.6, 37.0, 89.0
Snow, 0.1 m	273 to 253	0.25	0.1	1.4	10.7, 18.6, 37.0, 89.0

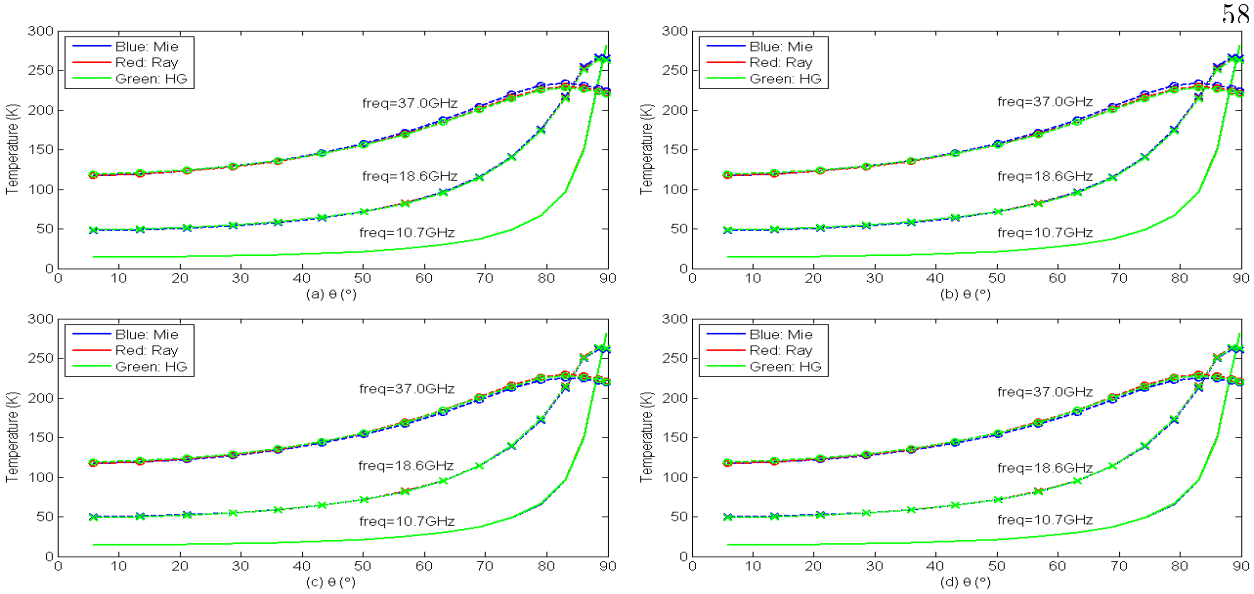


Figure 3.4: Brightness temperatures for a rain layer with a constant temperature profile: (a) horizontal-upwelling, (b) horizontal-downwelling, (c) vertical-upwelling, and (d) vertical-downwelling. Blue, red and green plots are made using the Mie, Rayleigh, and HG phase matrices, respectively.

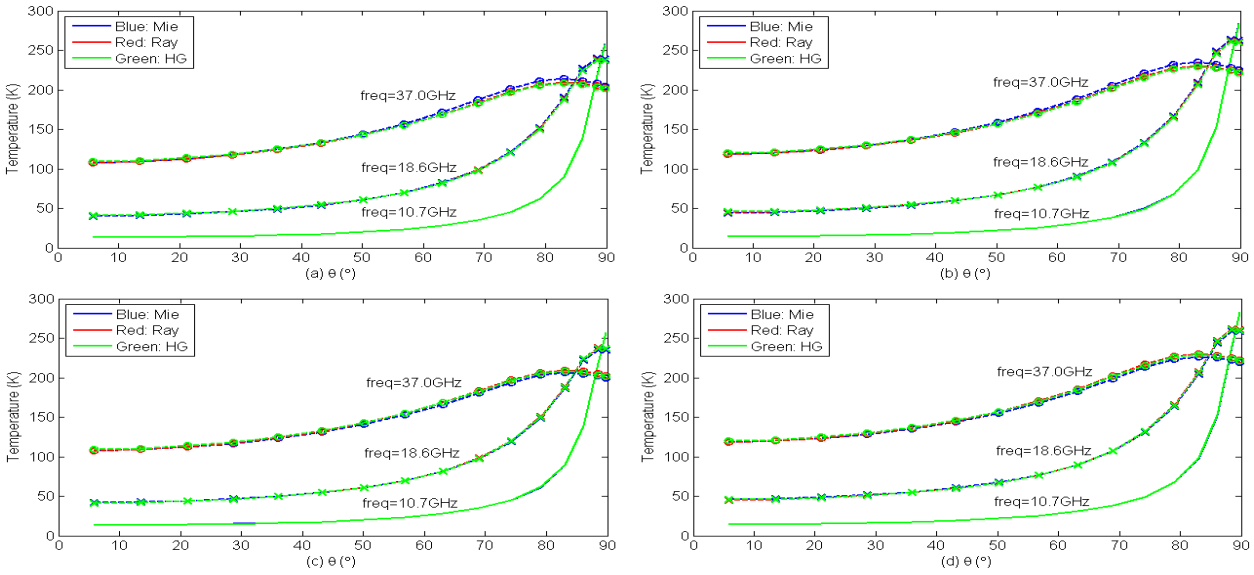


Figure 3.5: Brightness temperatures for a rain layer with a linear temperature profile: (a) horizontal-upwelling, (b) horizontal-downwelling, (c) vertical-upwelling, and (d) vertical-downwelling. Blue, red and green plots are made using the Mie, Rayleigh, and HG phase matrices, respectively.

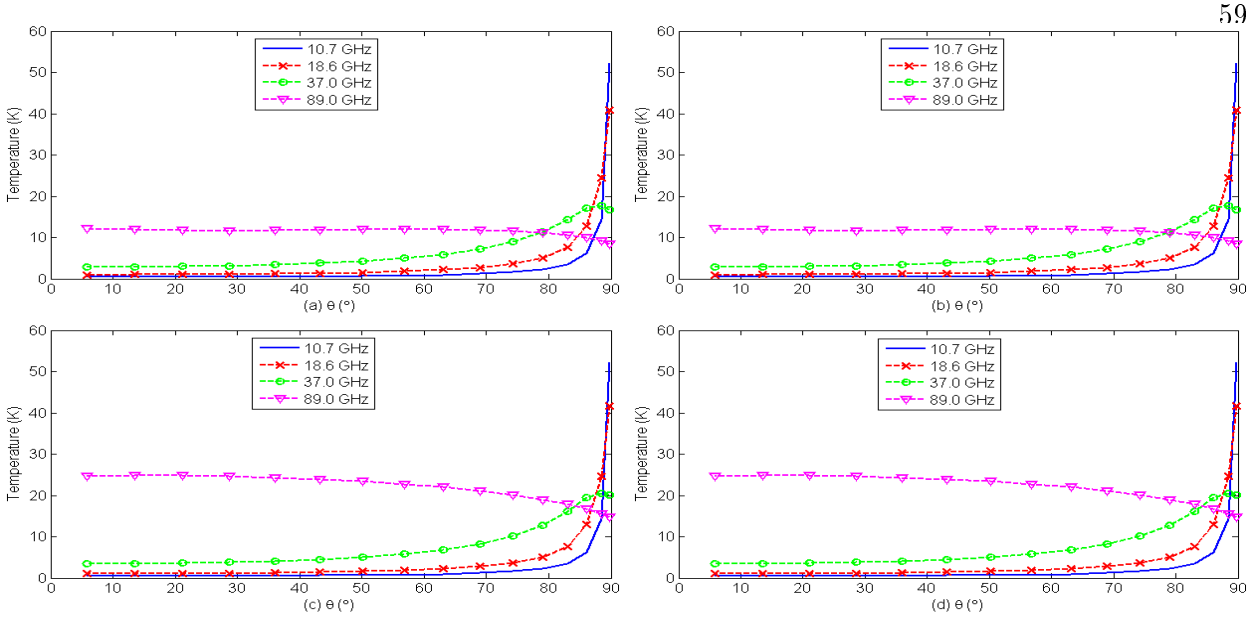


Figure 3.6: Brightness temperatures for a dry snow layer with a constant temperature profile: (a) horizontal-upwelling, (b) horizontal-downwelling, (c) vertical-upwelling, and (d) vertical-downwelling.

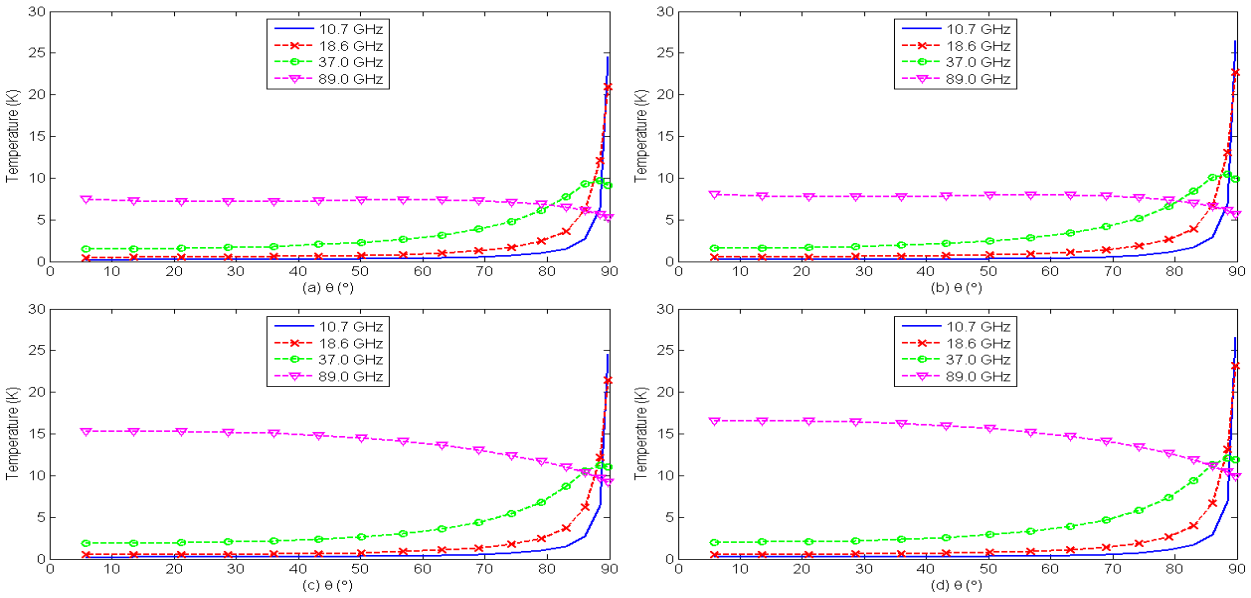


Figure 3.7: Brightness temperatures for a dry snow layer with a linear temperature profile: (a) horizontal-upwelling, (b) horizontal-downwelling, (c) vertical-upwelling, and (d) vertical-downwelling.

Chapter 4

Extended Jacobian Formulation of A Unified Microwave Radiative Transfer Model

4.1 Solutions for A Multilayer Stack with Refracting Layers

4.1.1 Refractivity Adjusted Reflection and Transmission Matrices for a Two-Layer Stack

For the case of a two-layer stack with a refracting boundary, the solutions for the refractivity adjusted reflection and transmission matrices are similar to those of the non-refracting case but with additional steps necessary to accommodate the Fresnel multistream reflection and refraction. Only a few of these key steps are provided since the solution is similar to that of Fung [42].

For the multiple reflection process illustrated in Fig. 4.1 the relevant reflection and transmission matrices are organized as two pairs of operators $\overline{\overline{R}}_{hi}^{\uparrow\downarrow}$ and $\overline{\overline{T}}_{hi}^{\uparrow\downarrow}$ and computed by considering the successive reflections that occur between the bulk volume scattering layer (layer 1) and upper homogeneous half space. This upper half space, which is presumably of a different effective permittivity than that of layer 1, is modeled by a Fresnel-Snell transition layer (layer 2, above the volume scattering layer of interest) that accounts for both the Fresnel reflection and transmission of streams and the refractive bending of streams and associated divergence of radiation according to Snell's law at the interface. The Fresnel-Snell layer reflectivity and transmissivity operators $\overline{\overline{r}}^{\uparrow\downarrow}$ and $\overline{\overline{t}}^{\uparrow\downarrow}$ (respectively) are annotated with arrows $\uparrow$ and $\downarrow$ to indicate the corresponding directions of the incident streams. The incident streams are the upwelling self-radiation from the bulk layer below, $\overline{u}_*$ (Fig. 4.1(a)) and the external downwelling radiation streams from the upper-half environment,

$\overline{v}_{inc}$ (Fig. 4.1(b)). Since all layer interfaces are specular the $\overline{\overline{r}}^{\uparrow\downarrow}$ operators are diagonal matrices and the $\overline{\overline{t}}^{\uparrow\downarrow}$ operators are banded non-diagonal matrices that described the interpolated stream angle bending. Note that for rough interfaces these matrices would generally all be full matrices. In this manner $\overline{\overline{r}}^{\uparrow\downarrow}$ and $\overline{\overline{t}}^{\uparrow\downarrow}$ properly compensate the angular streams by excluding the transmission beyond the critical angle (as relevant) and by applying either a linear or cubic spline interpolation.

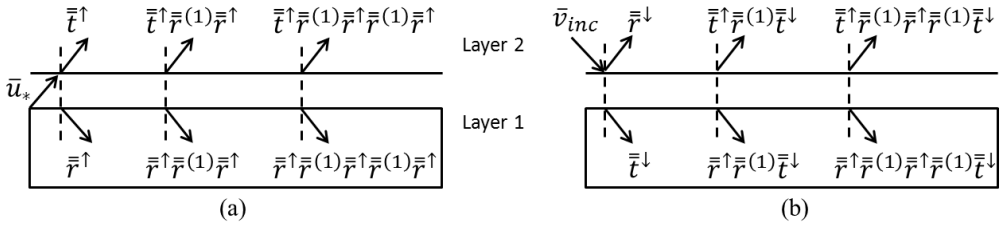


Figure 4.1: The scattering processes at the top surface of the intrinsic medium layer, assuming incident streams: (a) upwelling (↑) self-radiation streams $\overline{u}_*$, and (b) downwelling (↓) external incident streams $\overline{v}_{inc}$.

The use of the Fresnel-Snell operators is illustrated in Fig. 4.1 where the refractivity adjusted reflection and transmission matrices $\overline{\overline{R}}_{hi}^{\uparrow\downarrow}$ and $\overline{\overline{T}}_{hi}^{\uparrow\downarrow}$ between the Fresnel-Snell layer and the bulk volume scattering layer (whose reflection matrix is denoted by $\overline{\overline{r}}^{(1)}$) are defined by:

$$\begin{aligned} \overline{u}\,(z=h^+) &= \overline{\overline{T}}_{hi}^{\uparrow}\overline{u}_*\,(z=h^-) + \overline{\overline{R}}_{hi}^{\downarrow}\overline{v}_{inc}\,(z=h^+) \\ \overline{v}\,(z=h^-) &= \overline{\overline{T}}_{hi}^{\downarrow}\overline{v}_{inc}\,(z=h^+) + \overline{\overline{R}}_{hi}^{\uparrow}\overline{u}_*\,(z=h^-) \end{aligned} \tag{4.1}$$

where the superscripts + and − indicate evaluation at a height immediately above or below (respectively) the interface at $z = h$. Owing to multiple surface and volume reflections these refractivity adjusted matrices (denoted by subscript "hi", for a **h**omogeneous medium above an **i**ntrinsic medium layer) are computed as

$$
\begin{aligned}
\overline{\overline{R}}_{hi}^{\uparrow} &= \overline{\overline{r}}^{\uparrow}\left(\overline{\overline{I}} - \overline{\overline{r}}^{(1)}\overline{\overline{r}}^{\uparrow}\right)^{-1} \\
\overline{\overline{T}}_{hi}^{\uparrow} &= \overline{\overline{t}}^{\uparrow}\left(\overline{\overline{I}} - \overline{\overline{r}}^{(1)}\overline{\overline{r}}^{\uparrow}\right)^{-1} \\
\overline{\overline{R}}_{hi}^{\downarrow} &= \overline{\overline{r}}^{\downarrow} + \overline{\overline{t}}^{\uparrow}\overline{\overline{r}}^{(1)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1} \\
\overline{\overline{T}}_{hi}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1}\overline{\overline{t}}^{\downarrow}
\end{aligned}
\tag{4.2}
$$

Similarly, for a homogeneous medium underneath an intrinsic medium layer (denoted by subscript "ih", Fig. 4.2) another four refractivity adjusted matrices are computed:

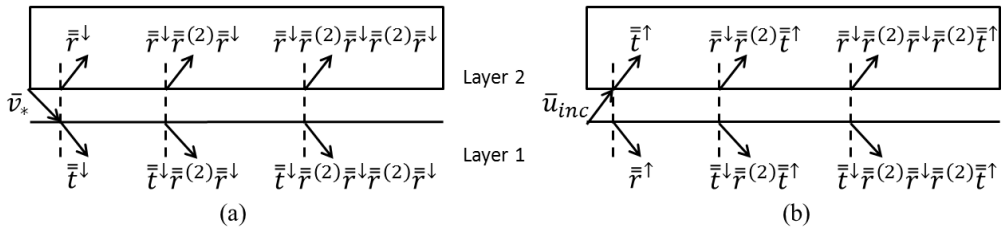


Figure 4.2: The scattering processes at the bottom surface of the intrinsic medium layer, assuming incident streams: (a) downwelling (↓) self-radiation streams $\overline{v}_*$, and (b) upwelling (↑) external incident streams $\overline{u}_{inc}$.

$$
\begin{aligned}
\overline{\overline{R}}_{ih}^{\downarrow} &= \overline{\overline{r}}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{r}}^{(2)}\overline{\overline{r}}^{\downarrow}\right)^{-1} \\
\overline{\overline{T}}_{ih}^{\downarrow} &= \overline{\overline{t}}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{r}}^{(2)}\overline{\overline{r}}^{\downarrow}\right)^{-1} \\
\overline{\overline{R}}_{ih}^{\uparrow} &= \overline{\overline{r}}^{\uparrow} + \overline{\overline{t}}^{\downarrow}\overline{\overline{r}}^{(2)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1} \\
\overline{\overline{T}}_{ih}^{\uparrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1}\overline{\overline{t}}^{\uparrow}
\end{aligned}
\tag{4.3}
$$

and applied to the following stream relationships:

$$
\begin{aligned}
\overline{u}\left(z = h^{+}\right) &= \overline{\overline{T}}_{ih}^{\uparrow}\overline{u}_{inc}\left(z = h^{-}\right) + \overline{\overline{R}}_{ih}^{\downarrow}\overline{v}_{*}\left(z = h^{+}\right) \\
\overline{v}\left(z = h^{-}\right) &= \overline{\overline{T}}_{ih}^{\downarrow}\overline{v}_{*}\left(z = h^{+}\right) + \overline{\overline{R}}_{ih}^{\uparrow}\overline{u}_{inc}\left(z = h^{-}\right)
\end{aligned}
\tag{4.4}
$$

Finally, the refractivity adjusted reflection and transmission matrices for two adjacent intrinsic medium layers can now be determined by introducing eight new downwelling (↓) multiple scattering operators, illustrated in Fig. 4.3 and defined by

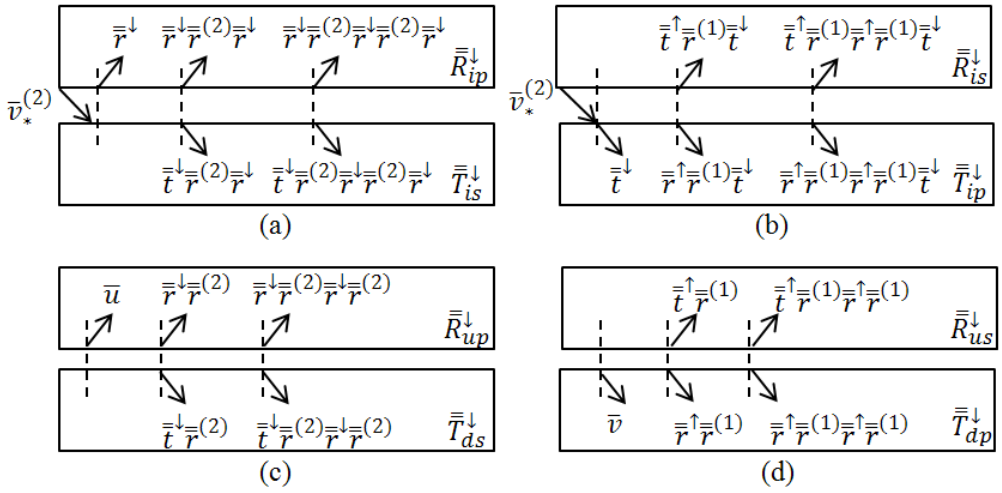


Figure 4.3: The scattering processes at the interface between two intrinsic medium layers due to the downwelling incident field, $\overline{v}_*^{(2)}$. (a-b): external incident source being $\overline{v}_*^{(2)}$, (c-d) internal incident sources: up- and down- welling due to $\overline{v}_*^{(2)}$. Eight multiple scattering operators are illustrated here.

$$
\begin{aligned}
\overline{\overline{R}}_{ip}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1}\overline{\overline{r}}^{\downarrow} \\
\overline{\overline{T}}_{is}^{\downarrow} &= \overline{\overline{t}}^{\downarrow}\overline{\overline{r}}^{(2)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1}\overline{\overline{r}}^{\downarrow} \\
\overline{\overline{R}}_{is}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\overline{\overline{r}}^{(1)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1}\overline{\overline{t}}^{\downarrow} \\
\overline{\overline{T}}_{ip}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1}\overline{\overline{t}}^{\downarrow} \\
\overline{\overline{R}}_{up}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1} \\
\overline{\overline{T}}_{ds}^{\downarrow} &= \overline{\overline{t}}^{\downarrow}\overline{\overline{r}}^{(2)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(2)}\right)^{-1} \\
\overline{\overline{R}}_{us}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\overline{\overline{r}}^{(1)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1} \\
\overline{\overline{T}}_{dp}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(1)}\right)^{-1}
\end{aligned}
\tag{4.5}
$$

When the incident radiation is from below $\overline{u}_*^{(1)}$, the corresponding eight upwelling (i.e., $\uparrow$) reflection and transmission operators are obtained in an analogous manner. The operators in (4.5) are applied to model the different but relevant multiple scattering processes at the intrinsic-intrinsic (denoted by subscript "ii") interface caused by the downwelling self-radiation streams $\overline{v}_*^{(2)}$ from layer 2. The relevant stream relationships for the **ii** case are:

$$\begin{aligned}\overline{u}\left(z=h^{+}\right) &= \overline{\overline{T}}_{ii}^{\uparrow}\overline{u}_{*}^{(1)}\left(z=h^{-}\right)+\overline{\overline{R}}_{ii}^{\downarrow}\overline{v}_{*}^{(2)}\left(z=h^{+}\right)\\ \overline{v}\left(z=h^{-}\right) &= \overline{\overline{T}}_{ii}^{\downarrow}\overline{v}_{*}^{(2)}\left(z=h^{+}\right)+\overline{\overline{R}}_{ii}^{\uparrow}\overline{u}_{*}^{(1)}\left(z=h^{-}\right)\end{aligned} \tag{4.6}$$

The resulting refractivity adjusted reflection and transmission matrices for the upwelling and downwelling incident streams are:

$$\begin{aligned}\overline{\overline{R}}_{ii}^{\downarrow\uparrow} = \;& \overline{\overline{R}}_{ip}^{\downarrow\uparrow}+\overline{\overline{R}}_{up}^{\downarrow\uparrow}\left(\overline{\overline{I}}-\overline{\overline{R}}_{us}^{\downarrow\uparrow}\overline{\overline{T}}_{ds}^{\downarrow\uparrow}\right)^{-1}\overline{\overline{R}}_{is}^{\downarrow\uparrow}\\ &+\overline{\overline{R}}_{up}^{\downarrow\uparrow}\left(\overline{\overline{I}}-\overline{\overline{R}}_{us}^{\downarrow\uparrow}\overline{\overline{T}}_{ds}^{\downarrow\uparrow}\right)^{-1}\overline{\overline{R}}_{us}^{\downarrow\uparrow}\overline{\overline{T}}_{is}^{\downarrow\uparrow}\end{aligned} \tag{4.7a}$$

$$\begin{aligned}\overline{\overline{T}}_{ii}^{\downarrow\uparrow} = \;& \overline{\overline{T}}_{ip}^{\downarrow\uparrow}+\overline{\overline{T}}_{dp}^{\downarrow\uparrow}\left(\overline{\overline{I}}-\overline{\overline{T}}_{ds}^{\downarrow\uparrow}\overline{\overline{R}}_{us}^{\downarrow\uparrow}\right)^{-1}\overline{\overline{T}}_{is}^{\downarrow\uparrow}\\ &+\overline{\overline{T}}_{dp}^{\downarrow\uparrow}\left(\overline{\overline{I}}-\overline{\overline{T}}_{ds}^{\downarrow\uparrow}\overline{\overline{R}}_{us}^{\downarrow\uparrow}\right)^{-1}\overline{\overline{T}}_{ds}^{\downarrow\uparrow}\overline{\overline{R}}_{is}^{\downarrow\uparrow}\end{aligned} \tag{4.7b}$$

The mnemonic aid shown in Fig. 4.4 can be used to determine the order of the various operators in Eq. (4.7b) for the downwelling case; a similar mnemonic holds for the upwelling case.

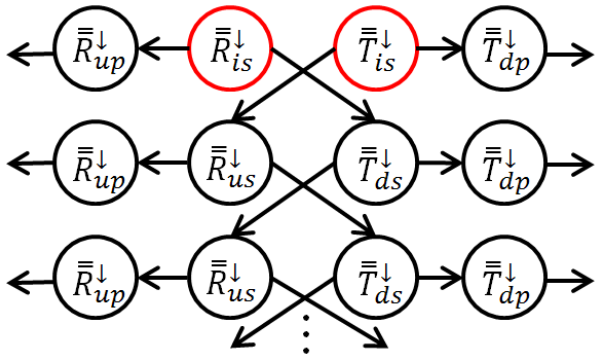
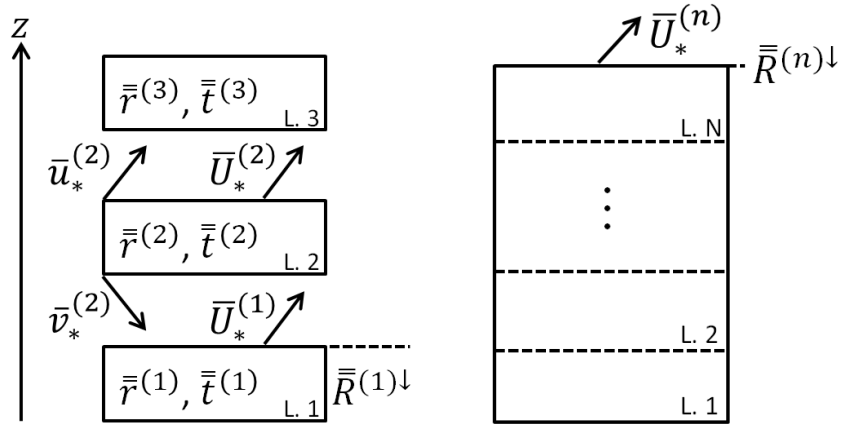


Figure 4.4: The reflected and transmitted radiation streams due to the multiple scattering between two intrinsic medium layers.

4.1.2 Solutions for a Multilayer Stack with Refracting Layers

The total refractivity adjusted scattering matrices and upwelling radiation streams of a multilayer stack are summarized here. Based on the upward recursive procedure outlined in [22, 42], the solution begins with a three-layer stack in which the third layer represents an upper homoge-

neous half space (for example, the atmosphere and above), as shown in Fig. 4.5(a). The first layer represents the lower half space.



(a) (b)

Figure 4.5: Illustration of the total emission, reflection, and transmission for a multilayer stack with refracting layers. (a) Three-layer stack with the third layer being background, (b) N-layer stack.

In Fig. 4.5(a), without including the contribution of the downwelling radiation from the background of the upper half space, the total upwelling radiation stream vector and refractivity adjusted scattering matrices of the two layers below can be calculated by applying Eqs. (4.1-4.7b) as follows:

$$\begin{aligned}
\bar{\bar{R}}^{(2)\downarrow} &= \bar{\bar{R}}^{\downarrow}_{hi32} + \bar{\bar{T}}^{\uparrow}_{hi23}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(2)}\bar{\bar{R}}^{(1)\downarrow}\bar{\bar{t}}^{(2)}\bar{\bar{R}}^{\uparrow}_{hi23}\right)^{-1} \\
&\quad \cdot\bar{\bar{t}}^{(2)}\bar{\bar{R}}^{(1)\downarrow}\bar{\bar{t}}^{(2)}\bar{\bar{T}}^{\downarrow}_{hi32} \\
\bar{\bar{T}}^{(2)\downarrow} &= \bar{\bar{T}}^{(1)\downarrow}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(2)}\bar{\bar{R}}^{\uparrow}_{hi23}\bar{\bar{t}}^{(2)}\bar{\bar{R}}^{(1)\downarrow}\right)^{-1}\bar{\bar{t}}^{(2)}\bar{\bar{T}}^{\downarrow}_{hi32} \\
\bar{U}_*^{(2)} &= \bar{\bar{T}}^{\uparrow}_{hi23}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(2)}\bar{\bar{R}}^{(1)\downarrow}\bar{\bar{t}}^{(2)}\bar{\bar{R}}^{\uparrow}_{hi23}\right)^{-1} \\
&\quad \cdot\left[\bar{u}_*^{(2)} + \bar{\bar{t}}^{(2)}\left(\bar{\bar{R}}^{(1)\downarrow}\bar{v}_*^{(2)} + \bar{U}_*^{(1)}\right)\right]
\end{aligned} \tag{4.8}$$

where $\bar{\bar{R}}^{(1)\downarrow}$ and $\bar{\bar{T}}^{(1)\downarrow}$ are the total refractivity adjusted reflection and transmission matrices evaluated at the top of layer 1, and (in this case) equivalent to $\bar{\bar{R}}^{\downarrow}_{ii21}$ and $\bar{\bar{T}}^{\downarrow}_{ii21}$, respectively.

If the effective permittivities of medium 1 and 2 are the same, which yields $\overline{\overline{r}}^{\uparrow\downarrow} = \overline{\overline{0}}$ and $\overline{\overline{t}}^{\uparrow\downarrow} = \overline{\overline{I}}$, then equations in (4.2) are simplified as:

$$\begin{aligned} \overline{\overline{R}}^{\uparrow}_{hi23} = \overline{\overline{0}}, \qquad \overline{\overline{R}}^{\downarrow}_{hi32} = \overline{\overline{r}}^{(2)} \\ \overline{\overline{T}}^{\uparrow}_{hi23} = \overline{\overline{T}}^{\downarrow}_{hi32} = \overline{\overline{I}} \end{aligned} \tag{4.9}$$

and Equations in (4.5) are simplified as

$$\begin{aligned} \overline{\overline{R}}^{\downarrow}_{ip} &= \overline{\overline{T}}^{\downarrow}_{is} = \overline{\overline{0}} \\ \overline{\overline{T}}^{\downarrow}_{ip} &= \overline{\overline{R}}^{\downarrow}_{up} = \overline{\overline{T}}^{\downarrow}_{dp} = \overline{\overline{I}} \\ \overline{\overline{R}}^{\downarrow}_{is} &= \overline{\overline{R}}^{\downarrow}_{us} = \overline{\overline{r}}^{(1)} \\ \overline{\overline{T}}^{\downarrow}_{ds} &= \overline{\overline{r}}^{(2)} \end{aligned} \tag{4.10}$$

Applying equations in (4.10) to Eq. (4.7a) yields

$$\overline{\overline{R}}^{\downarrow}_{ii21} = \left(\overline{\overline{I}} - \overline{\overline{r}}^{(1)}\overline{\overline{r}}^{(2)}\right)^{-1} \overline{\overline{r}}^{(1)} \tag{4.11}$$

Substituting Eqs. (4.9 and 4.11) into Eq. (4.8) yields

$$\overline{\overline{R}}^{(2)\downarrow} = \overline{\overline{r}}^{(2)} + \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{r}}^{(1)}\overline{\overline{r}}^{(2)}\right)^{-1} \overline{\overline{r}}^{(1)}\overline{\overline{t}}^{(2)} \tag{4.12}$$

, which is consistent with the expression in DOTLRT for the case of multilayer stack with non-refracting layers.

The overall recursion relations for an n-layer stack embedded within a taller N-layer stack are obtained as follows:

1) total refractivity adjusted reflection matrices:

$$
\begin{aligned}
\overline{\overline{R}}^{(1)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii21} + \overline{\overline{T}}^{\uparrow}_{ii12} \cdot \left(\overline{\overline{I}} - \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{(0)\downarrow} \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{\uparrow}_{ii12} \right)^{-1} \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{(0)\downarrow} \overline{\overline{t}}^{(1)} \overline{\overline{T}}^{\downarrow}_{ii21} \\
\vdots \quad & \quad \vdots \\
\overline{\overline{R}}^{(n-1)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii(n,n-1)} + \overline{\overline{T}}^{\uparrow}_{ii(n-1,n)} \cdot \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{(n-2)\downarrow} \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{\uparrow}_{ii(n-1,n)} \right)^{-1} \\
&\quad \cdot \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{(n-2)\downarrow} \overline{\overline{t}}^{(n-1)} \overline{\overline{T}}^{\downarrow}_{ii(n,n-1)} \\
\overline{\overline{R}}^{(n)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii(n+1,n)} + \overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \cdot \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{(n-1)\downarrow} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)} \right)^{-1} \\
&\quad \cdot \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{(n-1)\downarrow} \overline{\overline{t}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)}
\end{aligned}
\tag{4.13a}
$$

2) total refractivity adjusted transmission matrices

$$
\begin{aligned}
\overline{\overline{T}}^{(1)\downarrow} &= \overline{\overline{T}}^{(0)\downarrow} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{\uparrow}_{ii12} \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{(0)\downarrow} \right)^{-1} \overline{\overline{t}}^{(1)} \overline{\overline{T}}^{\downarrow}_{ii21} \\
\vdots \quad & \quad \vdots \\
\overline{\overline{T}}^{(n-1)\downarrow} &= \overline{\overline{T}}^{(n-2)\downarrow} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{\uparrow}_{ii(n-1,n)} \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{(n-2)\downarrow} \right)^{-1} \overline{\overline{t}}^{(n-1)} \overline{\overline{T}}^{\downarrow}_{ii(n,n-1)} \\
\overline{\overline{T}}^{(n)\downarrow} &= \overline{\overline{T}}^{(n-1)\downarrow} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{(n-1)\downarrow} \right)^{-1} \overline{\overline{t}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)}
\end{aligned}
\tag{4.13b}
$$

3) total upwelling radiation streams

$$
\begin{aligned}
\overline{U}^{(1)}_{*} &= \overline{\overline{T}}^{\uparrow}_{ii12} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{(0)\downarrow} \overline{\overline{t}}^{(1)} \overline{\overline{R}}^{\uparrow}_{ii12} \right)^{-1} \left[\overline{u}^{(1)}_{*} + \overline{\overline{t}}^{(1)} \left(\overline{\overline{R}}^{(0)\downarrow} \overline{v}^{(1)}_{*} + \overline{U}^{(0)}_{*} \right) \right] \\
\vdots \quad & \quad \vdots \\
\overline{U}^{(n-1)}_{*} &= \overline{\overline{T}}^{\uparrow}_{ii(n-1,n)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{(n-2)\downarrow} \overline{\overline{t}}^{(n-1)} \overline{\overline{R}}^{\uparrow}_{ii(n-1,n)} \right)^{-1} \\
&\quad \cdot \left[\overline{u}^{(n-1)}_{*} + \overline{\overline{t}}^{(n-1)} \left(\overline{\overline{R}}^{(n-2)\downarrow} \overline{v}^{(n-1)}_{*} + \overline{U}^{(n-2)}_{*} \right) \right] \\
\overline{U}^{(n)}_{*} &= \overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{(n-1)\downarrow} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)} \right)^{-1} \\
&\quad \cdot \left[\overline{u}^{(n)}_{*} + \overline{\overline{t}}^{(n)} \left(\overline{\overline{R}}^{(n-1)\downarrow} \overline{v}^{(n)}_{*} + \overline{U}^{(n-1)}_{*} \right) \right]
\end{aligned}
\tag{4.13c}
$$

where $\overline{\overline{R}}^{(0)\downarrow} = \overline{\overline{R}}^{\downarrow}_{ih10}$, $\overline{\overline{T}}^{(0)\downarrow} = \overline{\overline{T}}^{\downarrow}_{ih10}$, and $\overline{U}^{(0)}_{*} = \overline{u}_{s}$, which is the radiation from the lower homogeneous half space.

The above equations extend those of DOTLRT by accommodating linear temperature inhomogeneities within each medium layer and refractivity discontinuities between layers, and revert to

the DOTLRT equations in the relevant limits. Since the incident direction matters when imposing the concept of Fresnel reflectivity and transmissivity at layer interfaces and the UMRT Jacobian procedure requires both $\overline{\overline{R}}^{(n)}$ and $\overline{\overline{T}}^{(n)}$, the UMRT formulation explicitly uses these operators defined in terms of incident stream directions: $\overline{\overline{R}}^{(n)\uparrow\downarrow}$ and $\overline{\overline{T}}^{(n)\uparrow\downarrow}$. Noted that the operators, $\overline{\overline{R}}^{(n)\uparrow}$, $\overline{\overline{T}}^{(n)\uparrow}$, and $\overline{V}_*^{(n)}$ can be similarly calculated as in (4.13a) but using downward recursion.

It is finally noted that the above formulation of the total refractivity adjusted scattering matrices and upwelling radiation streams of a multilayer stack mainly follows from the derivation outlined in Chapter 10 of [42]. An equivalent formulation that follows the DOTLRT is provided in Appendix D.

4.2 Jacobian Procedure

4.2.1 Jacobian Solution for A Single Layer

One important feature of the UMRT model is the capability to rapidly and accurately compute the derivatives of the brightness temperature that would be observed at an arbitrary fixed height and angle with respect to any relevant radiative parameter, viz:

$$\frac{\partial T_B(\theta, z)}{\partial p_n}, \quad n = 0, 1, \cdots, N$$

where the parameter p_n can be any of κ_a, κ_s, T_o, γ, and d for the n^{th} layer. The procedure for obtaining these derivatives can be understood by first recalling that during recursion all quantities such as ($\overline{U}_*^{(n)}$, $\overline{V}_*^{(n)}$, $\overline{\overline{R}}^{(n)\uparrow\downarrow}$, $\overline{\overline{T}}^{(n)\uparrow\downarrow}$) and ($\overline{u}_*^{(n)}$, $\overline{v}_*^{(n)}$, $\overline{\overline{r}}^{(n)}$, $\overline{\overline{t}}^{(n)}$) are calculated and stored. Note that in UMRT (unlike in DOTLRT) the following inequalities generally hold: $\overline{u}_*^{(n)} \neq \overline{v}_*^{(n)}$, $\overline{\overline{R}}^{(n)\uparrow} \neq \overline{\overline{R}}^{(n)\downarrow}$, and $\overline{\overline{T}}^{(n)\uparrow} \neq \overline{\overline{T}}^{(n)\downarrow}$, and as such must be considered. If the n^{th} layer inside the N-layer stack is perturbed (call this the Jacobian layer), then the entire N-layer stack can be considered to be composed of three shorter stacks (sub-stacks) as illustrated in Fig. 4.6(a). As a result of the interaction of the radiation streams, $\overline{U}_*^{(n-1)}$, $\overline{V}_*^{(n+1)}$, $\overline{u}_*^{(n)}$, and $\overline{v}_*^{(n)}$ within these stacks at the bottom and top of the Jacobian layer will appear four newly defined upwelling and downwelling propagating

streams: $\overline{u}^{'}$, $\overline{v}^{'}$, $\overline{u}^{''}$, $\overline{v}^{''}$. These additional streams are expressed in

$$\begin{aligned}
\overline{u}^{'} &= \overline{\overline{R}}^{(n-1)\downarrow}\left(\overline{v}_{*}^{(n)}+\overline{v}^{'}\right)\\
\overline{v}^{'} &= \overline{\overline{r}}^{(n)}\left(\overline{U}_{*}^{(n-1)}+\overline{u}^{'}\right)+\overline{\overline{t}}^{(n)}\left(\overline{V}_{*}^{(n+1)}+\overline{v}^{''}\right)\\
\overline{u}^{''} &= \overline{\overline{t}}^{(n)}\left(\overline{U}_{*}^{(n-1)}+\overline{u}^{'}\right)+\overline{\overline{r}}^{(n)}\left(\overline{V}_{*}^{(n+1)}+\overline{v}^{''}\right)\\
\overline{v}^{''} &= \overline{\overline{R}}^{(n+1)\uparrow}\left(\overline{u}_{*}^{(n)}+\overline{u}^{''}\right)
\end{aligned} \tag{4.14}$$

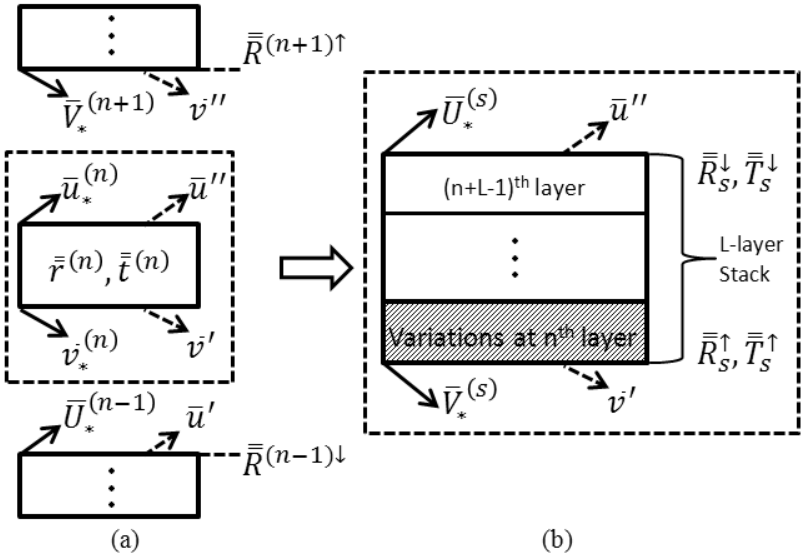


Figure 4.6: Schematic Jacobian calculation for observations at: (a) the perturbed layer (the n^{th} layer) and (b) an arbitrary level represented by an L-layer sub-stack in the middle of the entire stack.

A primary goal is to find the analytical derivatives of these streams, denoted as $\dot{\overline{u}}^{'}$, $\dot{\overline{v}}^{'}$, $\dot{\overline{u}}^{''}$, $\dot{\overline{v}}^{''}$ within the UMRT framework. An important advantage of this framework is that since only the n^{th} layer is perturbed, $\dot{\overline{u}}^{'}$, $\dot{\overline{v}}^{'}$, $\dot{\overline{u}}^{''}$, $\dot{\overline{v}}^{''}$ are only related to the variations in $\overline{u}_{*}^{(n)}$, $\overline{v}_{*}^{(n)}$, $\overline{\overline{r}}^{(n)}$, $\overline{\overline{t}}^{(n)}$, and are not affected by nor do they have an effect on $\overline{U}_{*}^{(n-1)}$, $\overline{V}_{*}^{(n+1)}$, $\overline{\overline{R}}^{(n-1)\downarrow}$, $\overline{\overline{R}}^{(n+1)\uparrow}$. Once the derivatives $\dot{\overline{u}}^{'}$, $\dot{\overline{v}}^{'}$, $\dot{\overline{u}}^{''}$, $\dot{\overline{v}}^{''}$ of the above streams for the Jacobian layer are calculated, the impact of them at each of the $n = 1, 2, \cdots N-1$ levels can be readily calculated by extending the middle stack from the level of the Jacobian layer to any arbitrary observation level, modeled by an L-layer

stack, $L = 1, 2, \cdots N-1$, shown in Fig. 4.6(b). For the L-layer stack of total reflection $\overline{\overline{R}}_s^{\uparrow\downarrow}$ and transmission $\overline{\overline{T}}_s^{\uparrow\downarrow}$ (as determined with matched upper and lower half spaces), Eq. (4.14) becomes:

$$\begin{aligned}
\overline{u}' &= \overline{\overline{R}}^{(n-1)\downarrow}\left(\overline{V}_*^{(s)} + \overline{v}'\right) \\
\overline{v}' &= \overline{\overline{R}}_s^{\uparrow}\left(\overline{U}_*^{(n-1)} + \overline{u}'\right) + \overline{\overline{T}}_s^{\downarrow}\left(\overline{V}_*^{(n+L)} + \overline{v}''\right) \\
\overline{u}'' &= \overline{\overline{T}}_s^{\uparrow}\left(\overline{U}_*^{(n-1)} + \overline{u}'\right) + \overline{\overline{R}}_s^{\downarrow}\left(\overline{V}_*^{(n+L)} + \overline{v}''\right) \\
\overline{v}'' &= \overline{\overline{R}}^{(n+L)\uparrow}\left(\overline{U}_*^{(s)} + \overline{u}''\right)
\end{aligned} \tag{4.15}$$

It is clear that Eq. (4.15) requires a similar calculation to solve for the streams and their derivatives compared to that of Eq. (4.14), and such calculation is straightforward. Hence, the basic problem is that of using Eq. (4.14) to obtain the derivative streams. The entire Jacobian procedure (Fig. 4.7) is thus an application of the derivative chain rule to Eq. (4.14) with all intermediate derivative matrices eventually found in terms of $\dot{\overline{\overline{A}}}$ and $\dot{\overline{\overline{B}}}$ [54]. Presented here are the key steps for differentiation with respect to the parameter p_n.

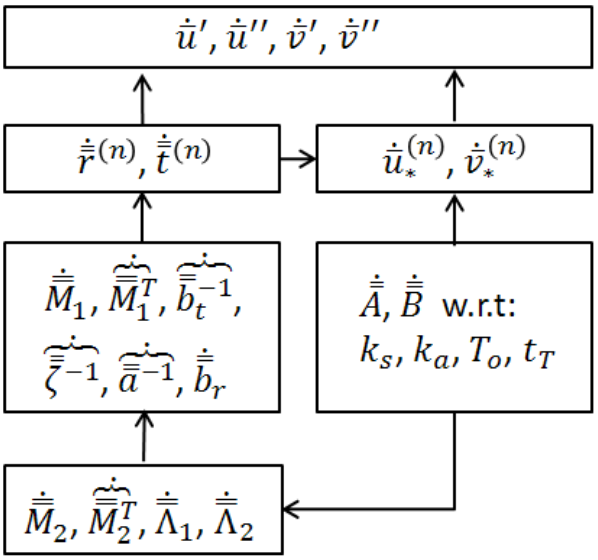


Figure 4.7: Flowchart of the UMRT Jacobian procedure.

4.2.1.1 Differentiation of the Four Newly Defined Propagating Streams

Following [22] the equations in (4.14) can be written:

$$\begin{aligned}\overline{u}'' &= \left(\overline{\overline{a}}_{11} - \overline{\overline{a}}_{12}\overline{\overline{a}}_{22}^{-1}\overline{\overline{a}}_{21}\right)^{-1}\left(\overline{b}_1 - \overline{\overline{a}}_{12}\overline{\overline{a}}_{22}^{-1}\overline{b}_2\right) \\ \overline{v}' &= \left(\overline{\overline{a}}_{22} - \overline{\overline{a}}_{21}\overline{\overline{a}}_{11}^{-1}\overline{\overline{a}}_{12}\right)^{-1}\left(\overline{b}_2 - \overline{\overline{a}}_{21}\overline{\overline{a}}_{11}^{-1}\overline{b}_1\right)\end{aligned} \tag{4.16a}$$

$$\begin{aligned}\overline{u}' &= \overline{\overline{R}}^{(n-1)\downarrow}\left(\overline{v}_*^{(n)} + \overline{v}'\right) \\ \overline{v}'' &= \overline{\overline{R}}^{(n+1)\uparrow}\left(\overline{u}_*^{(n)} + \overline{u}''\right)\end{aligned} \tag{4.16b}$$

where the intermediate variables are defined by

$$\begin{aligned}\overline{\overline{a}}_{11} &= \overline{\overline{I}} - \overline{\overline{r}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}, \quad & \overline{\overline{a}}_{12} &= -\overline{\overline{t}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow} \\ \overline{\overline{a}}_{21} &= -\overline{\overline{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}, & \overline{\overline{a}}_{22} &= \overline{\overline{I}} - \overline{\overline{r}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow}\end{aligned} \tag{4.17}$$

and

$$\begin{aligned}\overline{b}_1 = \; & \overline{\overline{r}}^{(n)}\left(\overline{V}_*^{(n+1)} + \overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)}\right) \\ & + \overline{\overline{t}}^{(n)}\left(\overline{U}_*^{(n-1)} + \overline{\overline{R}}^{(n-1)\downarrow}\overline{v}_*^{(n)}\right) \\ \overline{b}_2 = \; & \overline{\overline{t}}^{(n)}\left(\overline{V}_*^{(n+1)} + \overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)}\right) \\ & + \overline{\overline{r}}^{(n)}\left(\overline{U}_*^{(n-1)} + \overline{\overline{R}}^{(n-1)\downarrow}\overline{v}_*^{(n)}\right)\end{aligned} \tag{4.18}$$

Using these variables, $\overline{u}''$ and $\overline{v}'$ can be expressed in terms of $\overline{b}_1$ and $\overline{b}_2$ by the following equations

$$\begin{aligned}\overline{\overline{a}}_{11}\overline{u}'' + \overline{\overline{a}}_{12}\overline{v}' &= \overline{b}_1 \\ \overline{\overline{a}}_{21}\overline{u}'' + \overline{\overline{a}}_{22}\overline{v}' &= \overline{b}_2\end{aligned} \tag{4.19}$$

Differentiating (4.19) yields

$$\begin{aligned}\dot{\overline{\overline{a}}}_{11}\overline{u}'' + \overline{\overline{a}}_{11}\dot{\overline{u}}'' + \dot{\overline{\overline{a}}}_{12}\overline{v}' + \overline{\overline{a}}_{12}\dot{\overline{v}}' &= \dot{\overline{b}}_1 \\ \dot{\overline{\overline{a}}}_{21}\overline{u}'' + \overline{\overline{a}}_{21}\dot{\overline{u}}'' + \dot{\overline{\overline{a}}}_{22}\overline{v}' + \overline{\overline{a}}_{22}\dot{\overline{v}}' &= \dot{\overline{b}}_2\end{aligned} \tag{4.20}$$

The equations in (4.20) can be rearranged to be in the same form as (4.19):

$$\begin{aligned}\overline{\overline{a}}_{11}\dot{\overline{u}}'' + \overline{\overline{a}}_{12}\dot{\overline{v}}' &= \overline{c}_1 \\ \overline{\overline{a}}_{21}\dot{\overline{u}}'' + \overline{\overline{a}}_{22}\dot{\overline{v}}' &= \overline{c}_2\end{aligned} \tag{4.21}$$

where

$$\begin{aligned}\overline{c}_1 &\triangleq \dot{\overline{b}}_1 - \dot{\overline{\overline{a}}}_{11}\overline{u}'' - \dot{\overline{\overline{a}}}_{12}\overline{v}' \\ \overline{c}_2 &\triangleq \dot{\overline{b}}_2 - \dot{\overline{\overline{a}}}_{21}\overline{u}'' - \dot{\overline{\overline{a}}}_{22}\overline{v}'\end{aligned} \tag{4.22}$$

Hence, the derivatives of two of the four relevant streams are obtained as

$$\begin{aligned}\dot{\overline{u}}'' &= \left(\overline{\overline{a}}_{11} - \overline{\overline{a}}_{12}\overline{\overline{a}}_{22}^{-1}\overline{\overline{a}}_{21}\right)^{-1}\left(\overline{c}_1 - \overline{\overline{a}}_{12}\overline{\overline{a}}_{22}^{-1}\overline{c}_2\right) \\ \dot{\overline{v}}' &= \left(\overline{\overline{a}}_{22} - \overline{\overline{a}}_{21}\overline{\overline{a}}_{11}^{-1}\overline{\overline{a}}_{12}\right)^{-1}\left(\overline{c}_2 - \overline{\overline{a}}_{21}\overline{\overline{a}}_{11}^{-1}\overline{c}_1\right)\end{aligned} \tag{4.23}$$

Expressions for $\dot{\overline{\overline{a}}}_{ij}$ and $\dot{\overline{b}}_i$, $i,j = 1,2$ follow after differentiation of (4.17-4.18).

$$\begin{aligned}\dot{\overline{\overline{a}}}_{11} &= -\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow} & \dot{\overline{\overline{a}}}_{12} &= -\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow} \\ \dot{\overline{\overline{a}}}_{21} &= -\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow} & \dot{\overline{\overline{a}}}_{22} &= -\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow}\end{aligned} \tag{4.24}$$

$$\begin{aligned}\dot{\overline{b}}_1 = \;& \dot{\overline{\overline{r}}}^{(n)}\left(\overline{V}_*^{(n+1)} + \overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)}\right) + \overline{\overline{r}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\dot{\overline{u}}_*^{(n)} \\ & + \dot{\overline{\overline{t}}}^{(n)}\left(\overline{U}_*^{(n-1)} + \overline{\overline{R}}^{(n-1)\downarrow}\overline{v}_*^{(n)}\right) + \overline{\overline{t}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow}\dot{\overline{v}}_*^{(n)} \\ \dot{\overline{b}}_2 = \;& \dot{\overline{\overline{t}}}^{(n)}\left(\overline{V}_*^{(n+1)} + \overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)}\right) + \overline{\overline{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\dot{\overline{u}}_*^{(n)} \\ & + \dot{\overline{\overline{r}}}^{(n)}\left(\overline{U}_*^{(n-1)} + \overline{\overline{R}}^{(n-1)\downarrow}\overline{v}_*^{(n)}\right) + \overline{\overline{r}}^{(n)}\overline{\overline{R}}^{(n-1)\downarrow}\dot{\overline{v}}_*^{(n)}\end{aligned} \tag{4.25}$$

Finally, expressions for the other two streams $\dot{\overline{u}}'$ and $\dot{\overline{v}}''$ follow directly by differentiation of (4.16b) and application of (4.23).

$$\begin{aligned}\dot{\overline{u}}' &= \overline{\overline{R}}^{(n-1)\downarrow}\left(\dot{\overline{v}}_*^{(n)} + \dot{\overline{v}}'\right) \\ \dot{\overline{v}}'' &= \overline{\overline{R}}^{(n+1)\uparrow}\left(\dot{\overline{u}}_*^{(n)} + \dot{\overline{u}}''\right)\end{aligned} \tag{4.26}$$

4.2.1.2 Differentiation of $\overline{u}_*^{(n)}$ and $\overline{v}_*^{(n)}$

From (4.23-4.26) it is seen that the remaining problem is to find the four derivatives: $\dot{\overline{u}}_*^{(n)}$, $\dot{\overline{v}}_*^{(n)}$, $\dot{\overline{\overline{r}}}^{(n)}$, and $\dot{\overline{\overline{t}}}^{(n)}$. Assuming the thickness of the n^{th} layer is h, the self-radiation vectors $\overline{u}_*^{(n)}$ and $\overline{v}_*^{(n)}$ have the following explicit forms:

$$
\begin{aligned}
\overline{u}_*^{(n)} = & \underbrace{\left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\overline{\overline{A}}_n^{-1}\bar{F}_n(h)}_{P_{u1}} - \underbrace{\overline{\overline{t}}^{(n)}\overline{\overline{A}}_n^{-1}\bar{F}_n(0)}_{P_{u2}} \\
& + \underbrace{\left(\overline{\overline{I}} + \overline{\overline{r}}^{(n)} - \overline{\overline{t}}^{(n)}\right)\overline{\overline{B}}_n^{-1}\overline{\overline{A}}_n^{-1}\overline{\gamma}_T}_{P_{u3}} \\
\overline{v}_*^{(n)} = & \underbrace{\left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\overline{\overline{A}}_n^{-1}\bar{F}_n(0)}_{P_{v1}} - \underbrace{\overline{\overline{t}}^{(n)}\overline{\overline{A}}_n^{-1}\bar{F}_n(h)}_{P_{v2}} \\
& - \underbrace{\left(\overline{\overline{I}} + \overline{\overline{r}}^{(n)} - \overline{\overline{t}}^{(n)}\right)\overline{\overline{B}}_n^{-1}\overline{\overline{A}}_n^{-1}\overline{\gamma}_T}_{P_{v3}}
\end{aligned}
\tag{4.27}
$$

where subscript n denotes the n^{th} medium layer.

Using the symbols defined by the under-braces in (4.27), the derivatives $\dot{\overline{u}}_*^{(n)}$ and $\dot{\overline{v}}_*^{(n)}$ can now be represented as

$$
\begin{aligned}
\dot{\overline{u}}_*^{(n)} = & \quad \dot{P}_{u1} - \dot{P}_{u2} + \dot{P}_{u3} \\
\dot{\overline{v}}_*^{(n)} = & \quad \dot{P}_{v1} - \dot{P}_{v2} - \dot{P}_{v3}
\end{aligned}
\tag{4.28}
$$

Observing (4.27), it is seen that $\dot{\overline{u}}_*^{(n)}$ and $\dot{\overline{v}}_*^{(n)}$ are directly related to $\dot{\overline{\overline{A}}}_n^{-1}$, $\dot{\overline{\overline{B}}}_n^{-1}$, $\dot{\overline{\overline{r}}}^{(n)}$, $\dot{\overline{\overline{t}}}^{(n)}$, and $\dot{\bar{F}}_n$ by the product and chain rules. Moreover, the derivatives $\dot{\overline{\overline{r}}}^{(n)}$ and $\dot{\overline{\overline{t}}}^{(n)}$ are related to derivatives of $\overline{\overline{M}}_1$, $\overline{\overline{M}}_1^T$, $\overline{\overline{b}}_t^{-1}$, $\overline{\overline{b}}_r^{-1}$, $\overline{\overline{\zeta}}^{-1}$, and $\overline{\overline{a}}^{-1}$ for the n^{th} layer. All of these matrix operators are in turn directly related to $\overline{\overline{A}}_n$ and $\overline{\overline{B}}_n$.

For the n^{th} layer the matrices $\overline{\overline{A}}$ and $\overline{\overline{B}}$ have following explicit expressions [54]:

$$
\overline{\overline{A}} = \begin{bmatrix}
\frac{\kappa_e}{\mu_i}\delta_{ij} - \kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{vvij}^{++} + p_{vvij}^{+-}\right) & -\kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{vhij}^{++} + p_{vhij}^{+-}\right) \\
-\kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{hvij}^{++} + p_{hvij}^{+-}\right) & \frac{\kappa_e}{\mu_i}\delta_{ij} - \kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{hhij}^{++} + p_{hhij}^{+-}\right)
\end{bmatrix}
\tag{4.29}
$$

$$
\overline{\overline{B}} = \begin{bmatrix}
\frac{\kappa_e}{\mu_i}\delta_{ij} - \kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{vvij}^{++} - P_{vvij}^{+-}\right) & -\kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{vhij}^{++} - p_{vhij}^{+-}\right) \\
-\kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{hvij}^{++} - p_{hvij}^{+-}\right) & \frac{\kappa_e}{\mu_i}\delta_{ij} - \kappa_s\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p_{hhij}^{++} - p_{hhij}^{+-}\right)
\end{bmatrix}
$$

where $i, j = 1 \cdots M$, the symbol $p_{\alpha\beta ij}^{\pm\pm}$ represents corresponding reduced normalized phase matrix

elements. The linearized temperature profile for the n^{th} layer is $\overline{F} \triangleq \begin{bmatrix} \overline{f} \\ \\ \overline{f} \end{bmatrix}$, where $f_i = \sqrt{\frac{\gamma_i}{\mu_i}}\kappa_a(T_{\text{o}} - \gamma z)$, z$\in$ [0, h].

1) differentiating with respect to the scattering coefficient κ_s:

$$\dot{\overline{\overline{A}}} = \begin{bmatrix} \left\{\frac{\delta_{ij}}{\mu_i} - \sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{vvij} + p^{+-}_{vvij}\right)\right\} & \left\{-\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{vhij} + p^{+-}_{vhij}\right)\right\} \\ \left\{-\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{hvij} + p^{+-}_{hvij}\right)\right\} & \left\{\frac{\delta_{ij}}{\mu_i} - \sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{hhij} + p^{+-}_{hhij}\right)\right\} \end{bmatrix}$$

$$\dot{\overline{\overline{B}}} = \begin{bmatrix} \left\{\frac{\delta_{ij}}{\mu_i} - \sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{vvij} - p^{+-}_{vvij}\right)\right\} & \left\{-\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{vhij} - p^{+-}_{vhij}\right)\right\} \\ \left\{-\sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{hvij} - p^{+-}_{hvij}\right)\right\} & \left\{\frac{\delta_{ij}}{\mu_i} - \sqrt{\frac{\gamma_i\gamma_j}{\mu_i\mu_j}}\left(p^{++}_{hhij} - p^{+-}_{hhij}\right)\right\} \end{bmatrix} \tag{4.30}$$

and in this case

$$\dot{\overline{F}}_n = \overline{0} \tag{4.31}$$

Applying Eqs. (4.30) and (4.31) to Eq. (4.27) yields

$$\begin{aligned}
\dot{P}_{u1} = &\left[-\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{A}}_n^{-1} + \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\dot{\overline{\overline{A}}}_n^{-1}\right]\bar{F}_n(h) \\
\dot{P}_{u2} = &\left(\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{A}}_n^{-1} + \overline{\overline{t}}^{(n)}\dot{\overline{\overline{A}}}_n^{-1}\right)\bar{F}_n(0) \\
\dot{P}_{u3} = &\left(\dot{\overline{\overline{r}}}^{(n)} - \dot{\overline{\overline{t}}}^{(n)}\right)\overline{\overline{B}}_n^{-1}\overline{\overline{A}}_n^{-1}\overline{\gamma}_T + \left(\overline{\overline{I}} + \overline{\overline{r}}^{(n)} - \overline{\overline{t}}^{(n)}\right)\left(\dot{\overline{\overline{B}}}_n^{-1}\overline{\overline{A}}_n^{-1} + \overline{\overline{B}}_n^{-1}\dot{\overline{\overline{A}}}_n^{-1}\right)\overline{\gamma}_T \\
\dot{P}_{v1} = &\left[-\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{A}}_n^{-1} + \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\dot{\overline{\overline{A}}}_n^{-1}\right]\bar{F}_n(0) \\
\dot{P}_{v2} = &\left(\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{A}}_n^{-1} + \overline{\overline{t}}^{(n)}\dot{\overline{\overline{A}}}_n^{-1}\right)\bar{F}_n(h) \\
\dot{P}_{v3} = &\ \dot{P}_{u3}
\end{aligned} \tag{4.32}$$

2) differentiating with respect to the absorption coefficient κ_a:

$$\dot{\overline{\overline{A}}} = \dot{\overline{\overline{B}}} = \begin{bmatrix} \left\{\frac{1}{\mu_i}\delta_{ij}\right\} & \overline{\overline{0}} \\ \overline{\overline{0}} & \left\{\frac{1}{\mu_i}\delta_{ij}\right\} \end{bmatrix} \tag{4.33}$$

and

$$\begin{aligned} \dot{f}_{n_i}(h) = \sqrt{\tfrac{\gamma_i}{\mu_i}}(T_o - \gamma h) \quad &\Rightarrow \quad \dot{\overline{F}}_n(h) = \left[\dot{\overline{f}}_n(h), \quad \dot{\overline{f}}_n(h)\right]^T \\ \dot{f}_{n_i}(0) = \sqrt{\tfrac{\gamma_i}{\mu_i}}T_o \quad &\Rightarrow \quad \dot{\overline{F}}_n(0) = \left[\dot{\overline{f}}_n(0), \quad \dot{\overline{f}}_n(0)\right]^T \end{aligned} \tag{4.34}$$

Applying Eqs. (4.33) and (4.34) to Eq. (4.27) yields,

$$\begin{aligned} \dot{P}_{u1} = \quad & -\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{A}}_n^{-1}\bar{F}_n(h) + \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\left(\dot{\overline{\overline{A}}}_n^{-1}\bar{F}_n(h) + \overline{\overline{A}}_n^{-1}\dot{\bar{F}}_n(h)\right) \\ \dot{P}_{u2} = \quad & \left(\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{A}}_n^{-1} + \overline{\overline{t}}^{(n)}\dot{\overline{\overline{A}}}_n^{-1}\right)\bar{F}_n(0) + \overline{\overline{t}}^{(n)}\overline{\overline{A}}_n^{-1}\dot{\bar{F}}_n(0) \\ \dot{P}_{u3} = \quad & \left(\dot{\overline{\overline{r}}}^{(n)} - \dot{\overline{\overline{t}}}^{(n)}\right)\overline{\overline{B}}_n^{-1}\overline{\overline{A}}_n^{-1}\overline{\gamma}_T + \left(\overline{\overline{I}} + \overline{\overline{r}}^{(n)} - \overline{\overline{t}}^{(n)}\right)\left(\dot{\overline{\overline{B}}}_n^{-1}\overline{\overline{A}}_n^{-1} + \overline{\overline{B}}_n^{-1}\dot{\overline{\overline{A}}}_n^{-1}\right)\overline{\gamma}_T \\ \dot{P}_{v1} = \quad & -\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{A}}_n^{-1}\bar{F}_n(0) + \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right)\left(\dot{\overline{\overline{A}}}_n^{-1}\bar{F}_n(0) + \overline{\overline{A}}_n^{-1}\dot{\bar{F}}_n(0)\right) \\ \dot{P}_{v2} = \quad & \left(\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{A}}_n^{-1} + \overline{\overline{t}}^{(n)}\dot{\overline{\overline{A}}}_n^{-1}\right)\bar{F}_n(h) + \overline{\overline{t}}^{(n)}\overline{\overline{A}}_n^{-1}\dot{\bar{F}}_n(h) \\ \dot{P}_{v3} = \quad & \dot{P}_{u3} \end{aligned} \tag{4.35}$$

3) differentiating with respect to the thermodynamic temperature of medium T_o:

$$\dot{\overline{\overline{A}}} = \dot{\overline{\overline{B}}} = \overline{\overline{0}} \tag{4.36}$$

and

$$\dot{f}_{n_i}(h) = \dot{f}_{n_i}(0) = \sqrt{\tfrac{\gamma_i}{\mu_i}}\kappa_a \quad \Rightarrow \quad \dot{\overline{F}}_n = \left[\dot{\overline{f}}_n, \quad \dot{\overline{f}}_n\right]^T \tag{4.37}$$

Applying Eqs. (4.36) and (4.37) to Eq. (4.27) yields,

$$\begin{aligned}
\dot{P}_{u1} = \dot{P}_{v1} = &\quad \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right) \overline{\overline{A}}_n^{-1} \dot{\overline{F}}_n(h) \\
\dot{P}_{u2} = \dot{P}_{v2} = &\quad \overline{\overline{t}}^{(n)} \overline{\overline{A}}_n^{-1} \dot{\overline{F}}_n(0) \\
\dot{P}_{u3} = \dot{P}_{v3} = &\quad \overline{0}
\end{aligned} \tag{4.38}$$

4) differentiating with respect to the temperature lapse rate of the medium γ:

$$\dot{\overline{\overline{A}}} = \dot{\overline{\overline{B}}} = \overline{\overline{0}} \tag{4.39}$$

and

$$\begin{aligned}
\dot{f}_{n_i}(h) = \sqrt{\tfrac{\gamma_i}{\mu_i}} \kappa_a h \quad &\Rightarrow \quad \dot{\overline{F}}_n(h) = \left[\dot{\overline{f}}_n(h), \quad \dot{\overline{f}}_n(h)\right]^T \\
\dot{f}_{n_i}(0) = 0 \qquad\qquad &\Rightarrow \quad \dot{\overline{F}}_n(0) = \left[\dot{\overline{f}}_n(0), \quad \dot{\overline{f}}_n(0)\right]^T
\end{aligned} \tag{4.40}$$

Applying Eqs. (4.39) and (4.40) to Eq. (4.27) yields,

$$\begin{aligned}
\dot{P}_{u1} = &\quad \left(\overline{\overline{I}} - \overline{\overline{r}}^{(n)}\right) \overline{\overline{A}}_n^{-1} \dot{\overline{F}}_n(h) \\
\dot{P}_{u2} = &\quad \overline{0} \\
\dot{P}_{u3} = &\quad \left(\overline{\overline{I}} + \overline{\overline{r}}^{(n)} - \overline{\overline{t}}^{(n)}\right) \overline{\overline{B}}_n^{-1} \overline{\overline{A}}_n^{-1} \\
\dot{P}_{v1} = &\quad \overline{0} \\
\dot{P}_{v2} = &\quad \overline{\overline{t}}^{(n)} \overline{\overline{A}}_n^{-1} \dot{\overline{F}}_n(h) \\
\dot{P}_{v3} = &\quad \dot{P}_{u3}
\end{aligned} \tag{4.41}$$

5) differentiating with respect to the thickness of the medium d:

$$\dot{\overline{\overline{A}}} = \dot{\overline{\overline{B}}} = \overline{\overline{0}} \tag{4.42}$$

and

$$\begin{aligned}
\dot{f}_{n_i}(h) = \sqrt{\tfrac{\gamma_i}{\mu_i}} \kappa_a \gamma \quad &\Rightarrow \quad \dot{\overline{F}}_n(h) = \left[\dot{\overline{f}}_n(h), \quad \dot{\overline{f}}_n(h)\right]^T \\
\dot{f}_{n_i}(0) = 0 \qquad\qquad &\Rightarrow \quad \dot{\overline{F}}_n(0) = \left[\dot{\overline{f}}_n(0), \quad \dot{\overline{f}}_n(0)\right]^T
\end{aligned} \tag{4.43}$$

Applying Eqs. (4.42) and (4.43) to Eq. (4.27) yields,

$$
\begin{aligned}
\dot{P}_{u1} = & \; -\dot{\bar{\bar{r}}}^{(n)} \bar{\bar{A}}_n^{-1} \bar{F}_n(h) + \left(\bar{\bar{I}} - \bar{\bar{r}}^{(n)}\right) \bar{\bar{A}}_n^{-1} \dot{\bar{F}}_n(h) \\
\dot{P}_{u2} = & \; \dot{\bar{\bar{t}}}^{(n)} \bar{\bar{A}}_n^{-1} \bar{F}_n(0) \\
\dot{P}_{u3} = & \; \left(\dot{\bar{\bar{r}}}^{(n)} - \dot{\bar{\bar{t}}}^{(n)}\right) \bar{\bar{B}}_n^{-1} \bar{\bar{A}}_n^{-1} \bar{\gamma}_T \\
\dot{P}_{v1} = & \; -\dot{\bar{\bar{r}}}^{(n)} \bar{\bar{A}}_n^{-1} \bar{F}_n(0) \\
\dot{P}_{v2} = & \; \dot{\bar{\bar{t}}}^{(n)} \bar{\bar{A}}_n^{-1} \bar{F}_n(h) + \bar{\bar{t}}^{(n)} \bar{\bar{A}}_n^{-1} \dot{\bar{F}}_n(h) \\
\dot{P}_{v3} = & \; \dot{P}_{u3}
\end{aligned}
\tag{4.44}
$$

All $\dot{P}_{u1,2,3}$ and $\dot{P}_{v1,2,3}$ in Eqs. (4.32), (4.35), (4.38), and (4.41) are applied to Eq. (4.28) to calculate the derivatives $\dot{\bar{u}}_*^{(n)}$ and $\dot{\bar{v}}_*^{(n)}$, correspondingly.

4.2.1.3 Differentiation of $\bar{\bar{r}}^{(n)}$ and $\bar{\bar{t}}^{(n)}$

From Chapter 2, the reflection and transmission matrices for the n^{th} layer have following expressions:

$$
\bar{\bar{t}} = 2\bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{\zeta}}^{-1} \bar{\bar{a}}^{-1} \bar{\bar{M}}_1^T \tag{4.45}
$$

$$
\bar{\bar{r}} = \bar{\bar{t}} - \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{b}}_r \bar{\bar{M}}_1^T \tag{4.46}
$$

Directly applying the derivative chain rule to Eqs. (4.45) and (4.46), $\dot{\bar{\bar{r}}}^{(n)}$ and $\dot{\bar{\bar{t}}}^{(n)}$ can be expressed as

1-2) differentiating with respect to κ_s and κ_a:

$$
\begin{aligned}
\dot{\bar{\bar{r}}}^{(n)} = & \; \dot{\bar{\bar{t}}}^{(n)} - \dot{\bar{\bar{M}}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{b}}_r \bar{\bar{M}}_1^T - \bar{\bar{M}}_1 \overbrace{\bar{\bar{b}}_t^{-1}} \bar{\bar{b}}_r \bar{\bar{M}}_1^T \\
& - \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \dot{\bar{\bar{b}}}_r \bar{\bar{M}}_1^T - \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{b}}_r \overbrace{\bar{\bar{M}}_1^T}
\end{aligned}
\tag{4.47}
$$

$$
\begin{aligned}
\dot{\bar{\bar{t}}}^{(n)} = & \; 2(\dot{\bar{\bar{M}}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{\zeta}}^{-1} \bar{\bar{a}}^{-1} \bar{\bar{M}}_1^T + \bar{\bar{M}}_1 \overbrace{\bar{\bar{b}}_t^{-1}} \bar{\bar{\zeta}}^{-1} \bar{\bar{a}}^{-1} \bar{\bar{M}}_1^T \\
& + \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \overbrace{\bar{\bar{\zeta}}^{-1}} \bar{\bar{a}}^{-1} \bar{\bar{M}}_1^T + \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{\zeta}}^{-1} \overbrace{\bar{\bar{a}}^{-1}} \bar{\bar{M}}_1^T \\
& + \bar{\bar{M}}_1 \bar{\bar{b}}_t^{-1} \bar{\bar{\zeta}}^{-1} \bar{\bar{a}}^{-1} \overbrace{\bar{\bar{M}}_1^T})
\end{aligned}
\tag{4.48}
$$

3-4) differentiating with respect to T_o and γ:

$$\dot{\overline{\overline{r}}}^{(n)} = \dot{\overline{\overline{t}}}^{(n)} = \overline{\overline{0}} \tag{4.49}$$

5) differentiating with respect to d:

$$\dot{\overline{\overline{r}}}^{(n)} = \dot{\overline{\overline{t}}}^{(n)} - \overline{\overline{M}}_1 \dot{\overline{\overline{b}}}_t^{-1} \overline{\overline{b}}_r \overline{\overline{M}}_1^T - \overline{\overline{M}}_1 \overline{\overline{b}}_t^{-1} \dot{\overline{\overline{b}}}_r \overline{\overline{M}}_1^T \tag{4.50}$$

$$\dot{\overline{\overline{t}}}^{(n)} = 2\left(\overline{\overline{M}}_1 \dot{\overline{\overline{b}}}_t \overline{\overline{\zeta}}^{-1} \overline{\overline{a}}^{-1} \overline{\overline{M}}_1^T + \overline{\overline{M}}_1 \overline{\overline{b}}_t^{-1} \dot{\overline{\overline{\zeta}}}^{-1} \overline{\overline{a}}^{-1} \overline{\overline{M}}_1^T + \overline{\overline{M}}_1 \overline{\overline{b}}_t \overline{\overline{\zeta}}^{-1} \dot{\overline{\overline{a}}}^{-1} \overline{\overline{M}}_1^T \right) \tag{4.51}$$

The above details of the differentiation process are not presented in [22], however, the matrices in Eqs. (4.48) and (4.48) are well defined in [22] as

$$\begin{aligned}
\overline{\overline{\zeta}} &= \overline{\overline{\Lambda}}_2^{-\frac{1}{2}} \sinh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}} h \right) \\
\overline{\overline{a}} &= \overline{\overline{\Lambda}}_1^{\frac{1}{2}} \overline{\overline{M}}_2 + \overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \overline{\overline{M}}_2 \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \coth\left(\tfrac{1}{2} \overline{\overline{\Lambda}}_2^{\frac{1}{2}} h \right) \\
\overline{\overline{b}}_t &= \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{\frac{1}{2}} + \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \tanh\left(\tfrac{1}{2} \overline{\overline{\Lambda}}_2^{\frac{1}{2}} h \right) \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \\
\overline{\overline{b}}_r &= \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{\frac{1}{2}} - \overline{\overline{\Lambda}}_2^{\frac{1}{2}} \tanh\left(\tfrac{1}{2} \overline{\overline{\Lambda}}_2^{\frac{1}{2}} h \right) \overline{\overline{M}}_2^T \overline{\overline{\Lambda}}_1^{-\frac{1}{2}}
\end{aligned} \tag{4.52}$$

where $\overline{\overline{M}}_{1,2}$ are the orthogonal matrices consisting of eigenvectors from the symmetric matrices $\overline{\overline{A}}$ and $\overline{\overline{B}}$, respectively and $\overline{\overline{\Lambda}}_{1,2}$ are the associated diagonal matrices of corresponding eigenvalues $\lambda_{1,2(i)}$, defined by the following equations:

$$\begin{aligned}
\overline{\overline{A}} &= \overline{\overline{M}}_1 \overline{\overline{\Lambda}}_1 \overline{\overline{M}}_1^T \\
\overline{\overline{\Lambda}}_1^{\frac{1}{2}} \overline{\overline{M}}_1^T \overline{\overline{B}} \overline{\overline{M}}_1 \overline{\overline{\Lambda}}_1^{\frac{1}{2}} &= \overline{\overline{M}}_2 \overline{\overline{\Lambda}}_2 \overline{\overline{M}}_2^T
\end{aligned} \tag{4.53}$$

Note that in the above decomposition procedure $\overline{\overline{M}}_{1,2}^T = \overline{\overline{M}}_{1,2}^{-1}$. An important procedure in UMRT Jacobian is differentiation of $\overline{\overline{M}}_{1,2}$ and $\overline{\overline{\Lambda}}_{1,2}$ with respect to the relevant parameter of interest. This operation is equivalent to calculation of the derivatives of the eigenvalues and eigenvectors of a symmetric matrix, stated as

$$\begin{aligned}\dot{\overline{\overline{\Lambda}}}_1 &= \operatorname{diag}\left(\overline{\overline{M}}_{1T(n,n)}\right)\\ \dot{\overline{\overline{M}}}_{1(n,m)} &= \sum_{i\neq m}\frac{\overline{\overline{M}}_{1(n,i)}\overline{\overline{M}}_{1T(i,m)}}{\lambda_{1(m)}-\lambda_{1(i)}}\end{aligned} \tag{4.54}$$

where $\overline{\overline{M}}_{1T} = \overline{\overline{M}}_1^T\dot{\overline{\overline{A}}}\,\overline{\overline{M}}_1$ and the above result follows from first-order perturbation theory (e.g., Chapter 10 of [?] and Chapter 6 of [?]), which is introduced in Appendix C.

Similarly, letting $\overline{\overline{B}}_2 = \overline{\overline{\Lambda}}_1^{\frac{1}{2}}\overline{\overline{M}}_1^T\overline{\overline{B}}\,\overline{\overline{M}}_1\overline{\overline{\Lambda}}_1^{\frac{1}{2}} = \overline{\overline{M}}_2\overline{\overline{\Lambda}}_2\overline{\overline{M}}_2^T$, the derivative $\dot{\overline{\overline{B}}}_2$ can be obtained by

$$\begin{aligned}\dot{\overline{\overline{\Lambda}}}_2 &= \operatorname{diag}\left(\overline{\overline{M}}_{2T(n,n)}\right)\\ \dot{\overline{\overline{M}}}_{2(n,m)} &= \sum_{i\neq m}\frac{\overline{\overline{M}}_{2(n,i)}\overline{\overline{M}}_{2T(i,m)}}{\lambda_{2(m)}-\lambda_{2(i)}}\end{aligned} \tag{4.55}$$

where $\overline{\overline{M}}_{2T} = \overline{\overline{M}}_2^T\dot{\overline{\overline{B}}}_2\overline{\overline{M}}_2$.

An important procedure is also differentiation of an inverse matrix, $\overline{\overline{X}}^{-1}$. If the matrix $\overline{\overline{X}}$ is invertible and has derivative counterpart $\dot{\overline{\overline{X}}}$, then the following lemma holds:

$$\begin{aligned}&\overline{\overline{X}}\,\overline{\overline{X}}^{-1} = \overline{\overline{I}}\\ \Rightarrow\quad &\dot{\overbrace{\overline{\overline{X}}\,\overline{\overline{X}}^{-1}}} = \dot{\overline{\overline{X}}}\,\overline{\overline{X}}^{-1} + \overline{\overline{X}}\,\dot{\overbrace{\overline{\overline{X}}^{-1}}} = 0\\ \Rightarrow\quad &\dot{\overbrace{\overline{\overline{X}}^{-1}}} = -\overline{\overline{X}}^{-1}\dot{\overline{\overline{X}}}\,\overline{\overline{X}}^{-1}\end{aligned} \tag{4.56}$$

Applying the above lemma with known $\overline{\overline{A}}_n$, $\overline{\overline{B}}_n$, $\dot{\overline{\overline{M}}}_{1,2}$, and $\dot{\overline{\overline{\Lambda}}}_{1,2}$ matrices the differentiation of $\overline{\overline{A}}_n^{-1}$, $\overline{\overline{B}}_n^{-1}$, $\overline{\overline{M}}_{1,2}^T$, and $\overline{\overline{\Lambda}}_{1,2}^{-1}$ follows straightforwardly. This lemma is also applied to various scattering operators in forms such as $(\overline{\overline{I}} - \overline{\overline{R}}\overline{\overline{r}})^{-1}$, for example:

$$\begin{aligned}&\overline{\overline{R}}_p \triangleq \left(\overline{\overline{I}} - \overline{\overline{R}}\overline{\overline{r}}\right) \Rightarrow \dot{\overline{\overline{R}}}_p = -\dot{\overline{\overline{R}}}\overline{\overline{r}}\\ &\dot{\overbrace{\overline{\overline{R}}_p^{-1}}} = \left(\overline{\overline{I}} - \overline{\overline{R}}\overline{\overline{r}}\right)^{-1}\left(\dot{\overline{\overline{R}}}\overline{\overline{r}}\right)\left(\overline{\overline{I}} - \overline{\overline{R}}\overline{\overline{r}}\right)^{-1}\end{aligned} \tag{4.57}$$

Knowing $\dot{\overline{\overline{\zeta}}}$, $\dot{\overline{a}}$, $\dot{\overline{b}}_t$, and $\dot{\overline{b}}_r$ the corresponding derivatives of $\overline{\overline{\zeta}}^{-1}$, $\overline{a}^{-1}$, $\overline{b}_t^{-1}$, and $\overline{b}_r^{-1}$ are also readily found. For example, the derivative of $\overline{\overline{\zeta}}$ is

$$\begin{aligned}\dot{\overline{\overline{\zeta}}} &= \overbrace{\overline{\overline{\Lambda}}_2^{-\frac{1}{2}}}^{\cdot}\sinh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right) + \overline{\overline{\Lambda}}_2^{-\frac{1}{2}}\overbrace{\sinh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)}^{\cdot} \\ &= \overbrace{\overline{\overline{\Lambda}}_2^{-\frac{1}{2}}}^{\cdot}\sinh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right) + \overline{\overline{\Lambda}}_2^{-\frac{1}{2}}\cosh\left(\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\overbrace{\overline{\overline{\Lambda}}_2^{\frac{1}{2}}}^{\cdot}h\end{aligned} \tag{4.58}$$

where since $\overline{\overline{\Lambda}}_{1,2}$ are diagonal, $\overbrace{\overline{\overline{\Lambda}}_{1,2}^{-\frac{1}{2}}}^{\cdot} = -\frac{1}{2}\overline{\overline{\Lambda}}_{1,2}^{-\frac{3}{2}}\dot{\overline{\overline{\Lambda}}}_{1,2}$ and $\overbrace{\overline{\overline{\Lambda}}_{1,2}^{\frac{1}{2}}}^{\cdot} = \frac{1}{2}\overline{\overline{\Lambda}}_{1,2}^{-\frac{1}{2}}\dot{\overline{\overline{\Lambda}}}_{1,2}$. The other derivatives $\dot{\overline{\overline{a}}}$, $\dot{\overline{\overline{b}}}_t$ and $\dot{\overline{\overline{b}}}_r$ are

$$\begin{aligned}\dot{\overline{\overline{a}}} &= \dot{P}_{a1} + \dot{P}_{a2} + \dot{P}_{a3} + \dot{P}_{a4} + \dot{P}_{a5} \\ \dot{\overline{\overline{b}}}_t &= \dot{P}_{b1} + \dot{P}_{b2} + \dot{P}_{b3} + \dot{P}_{b4} + \dot{P}_{b5} \\ \dot{\overline{\overline{b}}}_r &= \dot{P}_{b1} - \dot{P}_{b2} - \dot{P}_{b3} - \dot{P}_{b4} - \dot{P}_{b5}\end{aligned} \tag{4.59}$$

where

$$\begin{aligned}\dot{P}_{a1} &= \overbrace{\overline{\overline{\Lambda}}_1^{\frac{1}{2}}\overline{\overline{M}}_2}^{\cdot} = \frac{1}{2}\overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\dot{\overline{\overline{\Lambda}}}_1\overline{\overline{M}}_2 + \overline{\overline{\Lambda}}_1^{\frac{1}{2}}\dot{\overline{\overline{M}}}_2 \\ \dot{P}_{a2} &= \left(-\frac{1}{2}\overline{\overline{\Lambda}}_1^{-\frac{3}{2}}\dot{\overline{\overline{\Lambda}}}_1\right)\overline{\overline{M}}_2\overline{\overline{\Lambda}}_2^{\frac{1}{2}}\coth\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right) \\ \dot{P}_{a3} &= \overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\dot{\overline{\overline{M}}}_2\overline{\overline{\Lambda}}_2^{\frac{1}{2}}\coth\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right) \\ \dot{P}_{a4} &= \overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\overline{\overline{M}}_2\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{-\frac{1}{2}}\dot{\overline{\overline{\Lambda}}}_2\right)\coth\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right) \\ \dot{P}_{a5} &= -\frac{1}{2}\overline{\overline{\Lambda}}_1^{-\frac{1}{2}}\overline{\overline{M}}_2\overline{\overline{\Lambda}}_2^{\frac{1}{2}}\operatorname{csch}^2\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\overbrace{\overline{\overline{\Lambda}}_2^{\frac{1}{2}}}^{\cdot}h \\ \dot{P}_{b1} &= \overbrace{\overline{\overline{M}}_2^T\overline{\overline{\Lambda}}_1^{\frac{1}{2}}}^{\cdot} = \overbrace{\overline{\overline{M}}_2^T}^{\cdot}\overline{\overline{\Lambda}}_1^{\frac{1}{2}} + \overline{\overline{M}}_2^T\overbrace{\overline{\overline{\Lambda}}_1^{\frac{1}{2}}}^{\cdot} \\ \dot{P}_{b2} &= \left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{-\frac{1}{2}}\dot{\overline{\overline{\Lambda}}}_2\right)\tanh\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\overline{\overline{M}}_2^T\overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \\ \dot{P}_{b3} &= \frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}\operatorname{sech}^2\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\left(\overbrace{\overline{\overline{\Lambda}}_2^{\frac{1}{2}}}^{\cdot}h\right)\overline{\overline{M}}_2^T\overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \\ \dot{P}_{b4} &= \overline{\overline{\Lambda}}_2^{\frac{1}{2}}\tanh\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\overbrace{\overline{\overline{M}}_2^T}^{\cdot}\overline{\overline{\Lambda}}_1^{-\frac{1}{2}} \\ \dot{P}_{b5} &= \overline{\overline{\Lambda}}_2^{\frac{1}{2}}\tanh\left(\frac{1}{2}\overline{\overline{\Lambda}}_2^{\frac{1}{2}}h\right)\overline{\overline{M}}_2^T\left(-\frac{1}{2}\overline{\overline{\Lambda}}_1^{-\frac{3}{2}}\dot{\overline{\overline{\Lambda}}}_1\right)\end{aligned}$$

Through the above steps the derivatives $\dot{\overline{u}}'$, $\dot{\overline{v}}'$, $\dot{\overline{u}}''$, and $\dot{\overline{v}}''$ are rapidly computed since all operations are $2M \times 2M$ matrix multiplications and only performed for the Jacobian layer.

4.2.2 Extended Jacobian for the L-layer Middle Stack

When the stack is augmented it is required to propagate this derivative upward by replacing (4.14) with (4.15), in which the matrices $\overline{\overline{r}}^{(n)}$ and $\overline{\overline{t}}^{(n)}$ are correspondingly replaced by the stack counterparts: $\overline{\overline{R}}_s^{\uparrow\downarrow}$ and $\overline{\overline{T}}_s^{\uparrow\downarrow}$, and the vectors $\overline{u}_*^{(n)}$ and $\overline{v}_*^{(n)}$ are replaced by $\overline{U}_*^{(s)}$ and $\overline{V}_*^{(s)}$, respectively. By this process the middle stack is augmented upward to L layers while the top stack is reduced accordingly and the bottom stack remains unchanged.

1) calculating $\dot{\overline{\overline{R}}}_s^{\downarrow}$, $\dot{\overline{\overline{T}}}_s^{\downarrow}$, and $\dot{\overline{U}}_*^{(s)}$:

This is a upward recursion procedure from the perturbed Jacobian layer (n^{th}) to the top of the L-layer middle stack. The explicit expressions of $\overline{\overline{R}}_s^{\downarrow}$, $\overline{\overline{T}}_s^{\downarrow}$, and $\overline{U}_*^{(s)}$ can be obtained using Eqs. (4.13a-4.13c). A remaining nontrivial operation is differentiation of the initial conditions used to begin upward recursion for the middle stack. These initial conditions at the bottom of the middle stack are:

$$\begin{aligned}
\overline{\overline{R}}^{(n)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii(n+1,n)} + \overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\downarrow}_{ih(n,n-1)} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)}\right)^{-1} \\
&\quad \cdot \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\downarrow}_{ih(n,n-1)} \overline{\overline{t}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)} \\
\overline{\overline{T}}^{(n)\downarrow} &= \overline{\overline{T}}^{\downarrow}_{ih(n,n-1)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\downarrow}_{ih(n,n-1)}\right)^{-1} \overline{\overline{t}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)} \\
\overline{U}_*^{(n)} &= \overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\downarrow}_{ih(n,n-1)} \overline{\overline{t}}^{(n)} \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)}\right)^{-1} \\
&\quad \cdot \left[\overline{u}_*^{(n)} + \overline{\overline{t}}^{(n)} \left(\overline{\overline{R}}^{\downarrow}_{ih(n,n-1)} \overline{v}_*^{(n)} + \overline{u}_s\right)\right]
\end{aligned} \tag{4.60}$$

For the middle stack, the first layer is the Jacobian layer and the background is assumed to be neutral and homogeneous, meaning that $\overline{\overline{R}}^{\downarrow}_{ih(n,n-1)} = \overline{\overline{0}}$ and $\overline{\overline{T}}^{\downarrow}_{ih(n,n-1)} = \overline{\overline{I}}$, and the source term below the isolated middle stack $\overline{u}_s = \overline{0}$. Thus, (4.60) can be simplified as

$$\begin{aligned}
\overline{\overline{R}}^{(n)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii(n+1,n)} \\
\Rightarrow \quad \dot{\overline{\overline{R}}}^{(n)\downarrow} &= \dot{\overline{\overline{R}}}^{\downarrow}_{ii(n+1,n)}
\end{aligned} \tag{4.61}$$

$$\begin{aligned}
\overline{\overline{T}}^{(n)\downarrow} &= \overline{\overline{t}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)} \\
\Rightarrow \quad \dot{\overline{\overline{T}}}^{(n)\downarrow} &= \dot{\overline{\overline{t}}}^{(n)} \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)} + \overline{\overline{t}}^{(n)} \dot{\overline{\overline{T}}}^{\downarrow}_{ii(n+1,n)}
\end{aligned} \tag{4.62}$$

$$\begin{aligned} \overline{U}_*^{(n)} &= \overline{\overline{T}}_{ii(n,n+1)}^{\uparrow}\overline{u}_*^{(n)} \\ \Rightarrow \quad \dot{\overline{U}}_*^{(n)} &= \dot{\overline{\overline{T}}}_{ii(n,n+1)}^{\uparrow}\overline{u}_*^{(n)} + \overline{\overline{T}}_{ii(n,n+1)}^{\uparrow}\dot{\overline{u}}_*^{(n)} \end{aligned} \tag{4.63}$$

In above equations, $\overline{\overline{R}}_{ii(n+1,n)}^{\downarrow}$ and $\overline{\overline{T}}_{ii(n+1,n)}^{\downarrow}$ can be calculated (respectively) by applying Eqs. (4.7a) and (4.7b) along with definitions in Eq. (4.5), from where

$$\begin{aligned} \overline{\overline{R}}_{ip}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\right)^{-1}\overline{\overline{r}}^{\downarrow} & \Rightarrow \quad \dot{\overline{\overline{R}}}_{ip}^{\downarrow} &= \overline{\overline{0}} \\ \overline{\overline{T}}_{is}^{\downarrow} &= \overline{\overline{t}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\right)^{-1}\overline{\overline{r}}^{\downarrow} & \Rightarrow \quad \dot{\overline{\overline{T}}}_{is}^{\downarrow} &= \overline{\overline{0}} \\ \overline{\overline{R}}_{up}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\right)^{-1} & \Rightarrow \quad \dot{\overline{\overline{R}}}_{up}^{\downarrow} &= \overline{\overline{0}} \\ \overline{\overline{T}}_{ds}^{\downarrow} &= \overline{\overline{t}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\downarrow}\overline{\overline{r}}^{(n+1)}\right)^{-1} & \Rightarrow \quad \dot{\overline{\overline{T}}}_{ds}^{\downarrow} &= \overline{\overline{0}} \end{aligned} \tag{4.64}$$

For $\overline{\overline{R}}_{is}^{\downarrow}$, $\overline{\overline{T}}_{ip}^{\downarrow}$, $\overline{\overline{R}}_{us}^{\downarrow}$, and $\overline{\overline{T}}_{dp}^{\downarrow}$, let

$$\overline{\overline{R}}_{p,1} = \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right) \Rightarrow \dot{\overline{\overline{R}}}_{p,1} = -\overline{\overline{r}}^{\uparrow}\dot{\overline{\overline{r}}}^{(n)} \tag{4.65}$$

Applying the Lemma, Eqs. (4.56) and (4.57),

$$\overbrace{\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right)^{-1}}^{\cdot} = \overbrace{\overline{\overline{R}}_{p,1}^{-1}}^{\cdot} = -\overline{\overline{R}}_{p,1}^{-1}\left(\dot{\overline{\overline{R}}}_{p,1}\right)\overline{\overline{R}}_{p,1}^{-1} = \overline{\overline{R}}_{p,1}^{-1}\left(\overline{\overline{r}}^{\uparrow}\dot{\overline{\overline{r}}}^{(n)}\right)\overline{\overline{R}}_{p,1}^{-1} \tag{4.66}$$

Using Eq. (4.66) to $\overline{\overline{R}}_{is}^{\downarrow}$, $\overline{\overline{T}}_{ip}^{\downarrow}$, $\overline{\overline{R}}_{us}^{\downarrow}$, and $\overline{\overline{T}}_{dp}^{\downarrow}$ yields

$$\begin{aligned} \overline{\overline{R}}_{is}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\overline{\overline{r}}^{(n)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right)^{-1}\overline{\overline{t}}^{\downarrow} & \Rightarrow \quad \dot{\overline{\overline{R}}}_{is}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\left(\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{R}}_{p,1}^{-1} + \overline{\overline{r}}^{(n)}\overbrace{\overline{\overline{R}}_{p,1}^{-1}}^{\cdot}\right)\overline{\overline{t}}^{\downarrow} \\ \overline{\overline{T}}_{ip}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right)^{-1}\overline{\overline{t}}^{\downarrow} & \Rightarrow \quad \dot{\overline{\overline{T}}}_{ip}^{\downarrow} &= \overbrace{\overline{\overline{R}}_{p,1}^{-1}}^{\cdot}\overline{\overline{t}}^{\downarrow} \\ \overline{\overline{R}}_{us}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\overline{\overline{r}}^{(n)}\left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right)^{-1} & \Rightarrow \quad \dot{\overline{\overline{R}}}_{us}^{\downarrow} &= \overline{\overline{t}}^{\uparrow}\left(\dot{\overline{\overline{r}}}^{(n)}\overline{\overline{R}}_{p,1}^{-1} + \overline{\overline{r}}^{(n)}\overbrace{\overline{\overline{R}}_{p,1}^{-1}}^{\cdot}\right) \\ \overline{\overline{T}}_{dp}^{\downarrow} &= \left(\overline{\overline{I}} - \overline{\overline{r}}^{\uparrow}\overline{\overline{r}}^{(n)}\right)^{-1} & \Rightarrow \quad & \overbrace{\overline{\overline{R}}_{p,1}^{-1}}^{\cdot} \end{aligned} \tag{4.67}$$

Similarly, let

$$\overline{\overline{R}}_{pa} = \left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right) \tag{4.68}$$

$$
\begin{aligned}
\overline{\overline{R}}_{pa}^{-1} &= \left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)^{-1} \\
\dot{\overline{\overline{R}}}_{pa} &= -\dot{\overline{\overline{R}}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow} \\
\Rightarrow \dot{\overline{\overline{R}}}_{pa}^{-1} &= -\overline{\overline{R}}_{pa}^{-1}\dot{\overline{\overline{R}}}_{pa}\overline{\overline{R}}_{pa}^{-1} \\
&= \left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)^{-1}\left(\dot{\overline{\overline{R}}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)\left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)^{-1}
\end{aligned}
\tag{4.69}
$$

and

$$
\overline{\overline{R}}_{pb} = \left(\overline{\overline{I}} - \overline{\overline{T}}_{ds}^{\downarrow}\overline{\overline{R}}_{us}^{\downarrow}\right) \tag{4.70}
$$

$$
\begin{aligned}
\overline{\overline{R}}_{pb}^{-1} &= \left(\overline{\overline{I}} - \overline{\overline{T}}_{ds}^{\downarrow}\overline{\overline{R}}_{us}^{\downarrow}\right)^{-1} \\
\dot{\overline{\overline{R}}}_{pb} &= -\overline{\overline{T}}_{ds}^{\downarrow}\dot{\overline{\overline{R}}}_{us}^{\downarrow} \\
\Rightarrow \dot{\overline{\overline{R}}}_{pb}^{-1} &= -\overline{\overline{R}}_{pb}^{-1}\dot{\overline{\overline{R}}}_{pb}\overline{\overline{R}}_{pb}^{-1} \\
&= \overline{\overline{R}}_{pb}^{-1}\left(\overline{\overline{T}}_{ds}^{\downarrow}\dot{\overline{\overline{R}}}_{us}^{\downarrow}\right)\overline{\overline{R}}_{pb}^{-1}
\end{aligned}
\tag{4.71}
$$

Applying Eqs. (4.64-4.71) to $\overline{\overline{R}}_{ii(n+1,n)}^{\downarrow}$ and $\overline{\overline{T}}_{ii(n+1,n)}^{\downarrow}$ yields

$$
\begin{aligned}
\overline{\overline{R}}_{ii(n+1,n)}^{\downarrow} = \;& \overline{\overline{R}}_{ip}^{\downarrow} + \overline{\overline{R}}_{up}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)^{-1}\overline{\overline{R}}_{is}^{\downarrow} \\
& + \overline{\overline{R}}_{up}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{ds}^{\downarrow}\right)^{-1}\overline{\overline{R}}_{us}^{\downarrow}\overline{\overline{T}}_{is}^{\downarrow}
\end{aligned}
\tag{4.72}
$$

$$
\begin{aligned}
\Rightarrow \dot{\overline{\overline{R}}}_{ii(n+1,n)}^{\downarrow} = \;& \overline{\overline{R}}_{up}^{\downarrow}\left(\dot{\overline{\overline{R}}}_{pa}^{-1}\overline{\overline{R}}_{is}^{\downarrow} + \overline{\overline{R}}_{pb}^{-1}\dot{\overline{\overline{R}}}_{is}^{\downarrow}\right) \\
& + \overline{\overline{R}}_{up}^{\downarrow}\left(\dot{\overline{\overline{R}}}_{pa}^{-1}\overline{\overline{R}}_{us}^{\downarrow} + \overline{\overline{R}}_{pb}^{-1}\dot{\overline{\overline{R}}}_{us}^{\downarrow}\right)\overline{\overline{T}}_{is}^{\downarrow}
\end{aligned}
\tag{4.73}
$$

and

$$
\begin{aligned}
\overline{\overline{T}}_{ii(n+1,n)}^{\downarrow} = \;& \overline{\overline{T}}_{ip}^{\downarrow} + \overline{\overline{T}}_{dp}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{T}}_{ds}^{\downarrow}\overline{\overline{R}}_{us}^{\downarrow}\right)^{-1}\overline{\overline{T}}_{is}^{\downarrow} \\
& + \overline{\overline{T}}_{dp}^{\downarrow}\left(\overline{\overline{I}} - \overline{\overline{T}}_{ds}^{\downarrow}\overline{\overline{R}}_{us}^{\downarrow}\right)^{-1}\overline{\overline{T}}_{ds}^{\downarrow}\overline{\overline{R}}_{is}^{\downarrow}
\end{aligned}
\tag{4.74}
$$

$$\Rightarrow \dot{\overline{\overline{T}}}^{\downarrow}_{ii(n+1,n)} = \dot{\overline{\overline{T}}}^{\downarrow}_{ip} + \left(\dot{\overline{\overline{T}}}^{\downarrow}_{dp} \overline{\overline{R}}^{-1}_{pb} + \overline{\overline{T}}^{\downarrow}_{dp} \dot{\overline{\overline{R}}}^{-1}_{pb} \right) \left(\overline{\overline{T}}^{\downarrow}_{is} + \overline{\overline{T}}^{\downarrow}_{ds} \overline{\overline{R}}^{\downarrow}_{is} \right) + \overline{\overline{T}}^{\downarrow}_{dp} \overline{\overline{R}}^{-1}_{pb} \overline{\overline{T}}^{\downarrow}_{ds} \dot{\overline{\overline{R}}}^{\downarrow}_{is} \tag{4.75}$$

Derivative of $\overline{\overline{T}}^{\uparrow}_{ii(n,n+1)}$ can be calculated in an analogous manner.

For the $(n+1)^{\text{th}}$ layer, the upward recursive solutions are:

$$\begin{aligned}
\overline{\overline{R}}^{(n+1)\downarrow} &= \overline{\overline{R}}^{\downarrow}_{ii(n+2,n+1)} + \overline{\overline{T}}^{\uparrow}_{ii(n+1,n+2)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \right)^{-1} \\
&\quad \cdot \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{T}}^{\downarrow}_{ii(n+2,n+1)} \\
\overline{\overline{T}}^{(n+1)\downarrow} &= \overline{\overline{T}}^{(n)\downarrow} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \right)^{-1} \overline{\overline{t}}^{(n+1)} \overline{\overline{T}}^{\downarrow}_{ii(n+2,n+1)} \\
\overline{U}^{(n+1)}_{*} &= \overline{\overline{T}}^{\uparrow}_{ii(n+1,n+2)} \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \right)^{-1} \\
&\quad \cdot \left[\overline{u}^{(n+1)}_{*} + \overline{\overline{t}}^{(n+1)} \left(\overline{\overline{R}}^{(n)\downarrow} \overline{v}^{(n+1)}_{*} + \overline{U}^{(n)}_{*} \right) \right]
\end{aligned} \tag{4.76}$$

To obtain the derivatives $\dot{\overline{\overline{R}}}^{(n+1)\downarrow}$, $\dot{\overline{\overline{T}}}^{(n+1)\downarrow}$, and $\dot{\overline{U}}^{(n+1)}_{*}$, let

$$\overline{\overline{R}}_{pc} = \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \right) \tag{4.77}$$

$$\begin{aligned}
\overline{\overline{R}}^{-1}_{pc} &= \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \right)^{-1} \\
\dot{\overline{\overline{R}}}_{pc} &= -\overline{\overline{t}}^{(n+1)} \dot{\overline{\overline{R}}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \\
\Rightarrow \dot{\overline{\overline{R}}}^{-1}_{pc} &= -\overline{\overline{R}}^{-1}_{pc} \dot{\overline{\overline{R}}}_{pc} \overline{\overline{R}}^{-1}_{pc} \\
&= \overline{\overline{R}}^{-1}_{pc} \left(\overline{\overline{t}}^{(n+1)} \dot{\overline{\overline{R}}}^{(n)\downarrow} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \right) \overline{\overline{R}}^{-1}_{pc}
\end{aligned} \tag{4.78}$$

and

$$\overline{\overline{R}}_{pd} = \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)} \overline{\overline{t}}^{(n+1)} \overline{\overline{R}}^{(n)\downarrow} \right) \tag{4.79}$$

$$
\begin{aligned}
\overline{\overline{R}}_{pd}^{-1} &= \left(\overline{\overline{I}} - \overline{\overline{t}}^{(n+1)}\overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)}\overline{\overline{t}}^{(n+1)}\overline{\overline{R}}^{(n)\downarrow}\right)^{-1} \\
\dot{\overline{\overline{R}}}_{pd} &= -\overline{\overline{t}}^{(n+1)}\overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)}\overline{\overline{t}}^{(n+1)}\dot{\overline{\overline{R}}}^{(n)\downarrow} \\
\Rightarrow \dot{\overline{\overline{R}}}_{pd}^{-1} &= -\overline{\overline{R}}_{pd}^{-1}\dot{\overline{\overline{R}}}_{pd}\overline{\overline{R}}_{pd}^{-1} \\
&= \overline{\overline{R}}_{pd}^{-1}\left(\overline{\overline{t}}^{(n+1)}\overline{\overline{R}}^{\uparrow}_{ii(n+1,n+2)}\overline{\overline{t}}^{(n+1)}\dot{\overline{\overline{R}}}^{(n)\downarrow}\right)\overline{\overline{R}}_{pd}^{-1}
\end{aligned}
\tag{4.80}
$$

Substituting Eqs. (4.77-4.80) into Eq. (4.76) yields

$$
\begin{aligned}
\dot{\overline{\overline{R}}}^{(n+1)\downarrow} &= \overline{\overline{T}}^{\uparrow}_{ii(n+1,n+2)}\left(\dot{\overline{\overline{R}}}_{pc}^{-1}\overline{\overline{t}}^{(n+1)}\overline{\overline{R}}^{(n)\downarrow} + \overline{\overline{R}}_{pc}^{-1}\overline{\overline{t}}^{(n+1)}\dot{\overline{\overline{R}}}^{(n)\downarrow}\right)\overline{\overline{t}}^{(n+1)}\overline{\overline{T}}^{\downarrow}_{ii(n+2,n+1)} \\
\dot{\overline{\overline{T}}}^{(n+1)\downarrow} &= \left(\dot{\overline{\overline{T}}}^{(n)\downarrow}\overline{\overline{R}}_{pd}^{-1} + \overline{\overline{T}}^{(n)\downarrow}\dot{\overline{\overline{R}}}_{pd}^{-1}\right)\overline{\overline{t}}^{(n+1)}\overline{\overline{T}}^{\downarrow}_{ii(n+2,n+1)} \\
\dot{\overline{U}}_{*}^{(n+1)} &= \overline{\overline{T}}^{\uparrow}_{ii(n+1,n+2)}\dot{\overline{\overline{R}}}_{pc}^{-1}\left[\overline{u}_{*}^{(n+1)} + \overline{\overline{t}}^{(n+1)}\left(\overline{\overline{R}}^{(n)\downarrow}\overline{v}_{*}^{(n+1)} + \overline{U}_{*}^{(n)}\right)\right] \\
&\quad + \overline{\overline{T}}^{\uparrow}_{ii(n+1,n+2)}\overline{\overline{R}}_{pc}^{-1}\overline{\overline{t}}^{(n+1)}\left(\dot{\overline{\overline{R}}}^{(n)\downarrow}\overline{v}_{*}^{(n+1)} + \dot{\overline{U}}_{*}^{(n)}\right)
\end{aligned}
\tag{4.81}
$$

Then $\dot{\overline{\overline{R}}}_{s}^{\downarrow}$, $\dot{\overline{\overline{T}}}_{s}^{\downarrow}$, and $\dot{\overline{U}}_{*}^{(s)}$ can be calculated via the upward recursions, Eqs. (4.13a-4.13c) using an analogous procedure as in Eqs. (4.64-4.81).

2) calculating $\dot{\overline{\overline{R}}}_{s}^{\uparrow}$, $\dot{\overline{\overline{T}}}_{s}^{\uparrow}$, and $\dot{\overline{V}}_{*}^{(s)}$:

This is a downward recursion procedure from the top of the L-layer middle stack to the perturbed Jacobian layer (n^{th}). The explicit expressions of $\overline{\overline{R}}_{s}^{\uparrow}$, $\overline{\overline{T}}_{s}^{\uparrow}$, and $\overline{V}_{*}^{(s)}$ can be obtained using Eqs. (4.13a-4.13c). In this case the background (above the middle stack) is assumed to be neutral and homogeneous, meaning that $\overline{\overline{R}}^{\uparrow}_{hi(n+L-1,\,n+L)} = \overline{\overline{0}}$ and $\overline{\overline{T}}^{\uparrow}_{hi(n+L-1,\,n+L)} = \overline{\overline{I}}$, and the source term above the isolated middle stack $\overline{v}_{s} = \overline{0}$. Thus,

$$
\begin{aligned}
\overline{\overline{R}}^{(n+L-1)\uparrow} &= \overline{\overline{R}}^{\uparrow}_{ii(n+L-2,n+L-1)} \\
\overline{\overline{T}}^{(n+L-1)\uparrow} &= \bar{\bar{t}}^{(n+L-1)}\overline{\overline{T}}^{\uparrow}_{ii(n+L-2,n+L-1)} \\
\overline{V}_*^{(n+L-1)} &= \overline{\overline{T}}^{\downarrow}_{ii(n+L-1,n+L-2)}\overline{v}_*^{(n+L-1)} \\
\vdots & \quad \vdots \\
\overline{\overline{R}}^{(n+1)\uparrow} &= \overline{\overline{R}}^{\uparrow}_{ii(n,n+1)} + \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{(n+2)\uparrow}\bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{\downarrow}_{ii(n+1,n)}\right)^{-1} \\
&\quad \cdot \bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{(n+2)\uparrow}\bar{\bar{t}}^{(n+1)}\overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \\
\overline{\overline{T}}^{(n+1)\uparrow} &= \overline{\overline{T}}^{(n+2)\uparrow}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{\downarrow}_{ii(n+1,n)}\bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{(n+2)\uparrow}\right)^{-1}\bar{\bar{t}}^{(n+1)}\overline{\overline{T}}^{\uparrow}_{ii(n,n+1)} \\
\overline{V}_*^{(n+1)} &= \overline{\overline{T}}^{\downarrow}_{ii(n+1,n)}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{(n+2)\uparrow}\bar{\bar{t}}^{(n+1)}\overline{\overline{R}}^{\downarrow}_{ii(n+1,n)}\right)^{-1} \\
&\quad \cdot \left[\overline{v}_*^{(n+1)} + \bar{\bar{t}}^{(n+1)}\left(\overline{\overline{R}}^{(n+2)\uparrow}\overline{u}_*^{(n+1)} + \overline{V}_*^{(n+2)}\right)\right]
\end{aligned}
\tag{4.82}
$$

In above equation, the derivatives $\dot{\bar{\bar{R}}}^{\downarrow}_{ii(n+1,n)}$, $\dot{\bar{\bar{T}}}^{\downarrow}_{ii(n+1,n)}$, and $\dot{\bar{\bar{T}}}^{\uparrow}_{ii(n,n+1)}$ are solved in the previous upward procedure and $\overline{\overline{R}}^{\uparrow}_{ii(n,n+1)}$ can be differentiated using an analogous procedure as in Eqs. (4.64-4.75). Then derivatives $\dot{\overline{\overline{R}}}^{(n+1)\uparrow}$, $\dot{\overline{\overline{T}}}^{(n+1)\uparrow}$, and $\dot{\overline{V}}_*^{(n+1)}$ can be calculated using the procedure described in Eqs. (4.76-4.81). Applying these derivatives in the following equations:

$$
\begin{aligned}
\overline{\overline{R}}^{(s)\uparrow} \triangleq \overline{\overline{R}}^{(n)\uparrow} &= \overline{\overline{R}}^{\uparrow}_{ih(n-1,n)} + \overline{\overline{T}}^{\downarrow}_{ih(n,n-1)}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\bar{\bar{t}}^{(n)}\overline{\overline{R}}^{\downarrow}_{ih(n,n-1)}\right)^{-1}\bar{\bar{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\bar{\bar{t}}^{(n)}\overline{\overline{T}}^{\uparrow}_{ih(n-1,n)} \\
&= \bar{\bar{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\bar{\bar{t}}^{(n)} \\
\overline{\overline{T}}^{(s)\uparrow} \triangleq \overline{\overline{T}}^{(n)\uparrow} &= \overline{\overline{T}}^{(n+1)\uparrow}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n)}\overline{\overline{R}}^{\downarrow}_{ih(n,n-1)}\bar{\bar{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\right)^{-1}\bar{\bar{t}}^{(n)}\overline{\overline{T}}^{\uparrow}_{ih(n-1,n)} \\
&= \overline{\overline{T}}^{(n+1)\uparrow}\bar{\bar{t}}^{(n)} \\
\overline{V}_*^{(s)} \triangleq \overline{V}_*^{(n)} &= \overline{\overline{T}}^{\downarrow}_{ih(n,n-1)}\left(\bar{\bar{I}} - \bar{\bar{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\bar{\bar{t}}^{(n)}\overline{\overline{R}}^{\downarrow}_{ih(n,n-1)}\right)^{-1}\left[\overline{v}_*^{(n)} + \bar{\bar{t}}^{(n)}\left(\overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)} + \overline{V}_*^{(n+1)}\right)\right] \\
&= \overline{v}_*^{(n)} + \bar{\bar{t}}^{(n)}\left(\overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)} + \overline{V}_*^{(n+1)}\right)
\end{aligned}
\tag{4.83}
$$

Thus,

$$
\begin{aligned}
\dot{\overline{\overline{R}}}^{(s)\uparrow} &= \left(\dot{\overline{\overline{t}}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow} + \overline{\overline{t}}^{(n)}\dot{\overline{\overline{R}}}^{(n+1)\uparrow}\right)\overline{\overline{t}}^{(n)} + \overline{\overline{t}}^{(n)}\overline{\overline{R}}^{(n+1)\uparrow}\dot{\overline{\overline{t}}}^{(n)} \\
\dot{\overline{\overline{T}}}^{(s)\uparrow} &= \dot{\overline{\overline{T}}}^{(n+1)\uparrow}\overline{\overline{t}}^{(n)} + \overline{\overline{T}}^{(n+1)\uparrow}\dot{\overline{\overline{t}}}^{(n)} \\
\dot{\overline{V}}_*^{(s)} &= \dot{\overline{v}}_*^{(n)} + \overline{\overline{t}}^{(n)}\left(\overline{\overline{R}}^{(n+1)\uparrow}\overline{u}_*^{(n)} + \overline{V}_*^{(n+1)}\right) \\
&\quad + \overline{\overline{t}}^{(n)}\left(\dot{\overline{\overline{R}}}^{(n+1)\uparrow}\overline{u}_*^{(n)} + \overline{\overline{R}}^{(n+1)\uparrow}\dot{\overline{u}}_*^{(n)} + \dot{\overline{V}}_*^{(n+1)}\right)
\end{aligned}
\tag{4.84}
$$

Chapter 5

UMRT-Jacobian: Numerical Validation and Field Data Intercomparison

5.1 Numerical Validation

Several examples of the validation and utility of the UMRT procedures have been studied. First, the importance of compensation of the refracted radiation streams by excluding transmission beyond inter-layer critical angles (θ_c) is demonstrated. A cubic spline with a "not a knot" end condition [55] is applied to interpolate transmission coefficients from the refracted angles of Snell's law to the fixed UMRT quadrature angles (Fig. 5.1), where for purposes of illustration we use a large number ($n = 5000$) of incident radiation streams with the angles $\theta_1, \cdots \theta_n$. In Figs. 5.1(a-b) the radiation streams propagate from vacuum ($\varepsilon_1 = 1$) into an optically denser medium ($\varepsilon_2 = 2$), hence all incident streams pass through the boundary but are bent into a narrower cone of angles within [0, θ_c] (represented by blue circles). Applying cubic spline interpolation, the transmission at the $M = 16$ fixed quadrature angles is represented by red circles. As a result of the critical angle, transmission at quadrature angles greater than θ_c inside the denser medium are zero.

Conversely, in Figs. 5.1(c-d) the radiation streams propagating from the optically denser medium ($\varepsilon_1 = 2$) to vacuum with angles beyond θ_c are excluded from transmission and instead reflected. The remaining incident streams are transmitted according to Snell's and the Fresnel relations (represented by blue circles). Applying cubic spline interpolation, the estimated transmission includes all the quadrature angles spread over [0, $\frac{\pi}{2}$], represented by red circles in Figs. 5.1(c-d).

Validation of the UMRT solution is performed by testing for energy conservation under conditions of thermal equilibrium. If a single layer or multilayer stack has the same thermodynamic

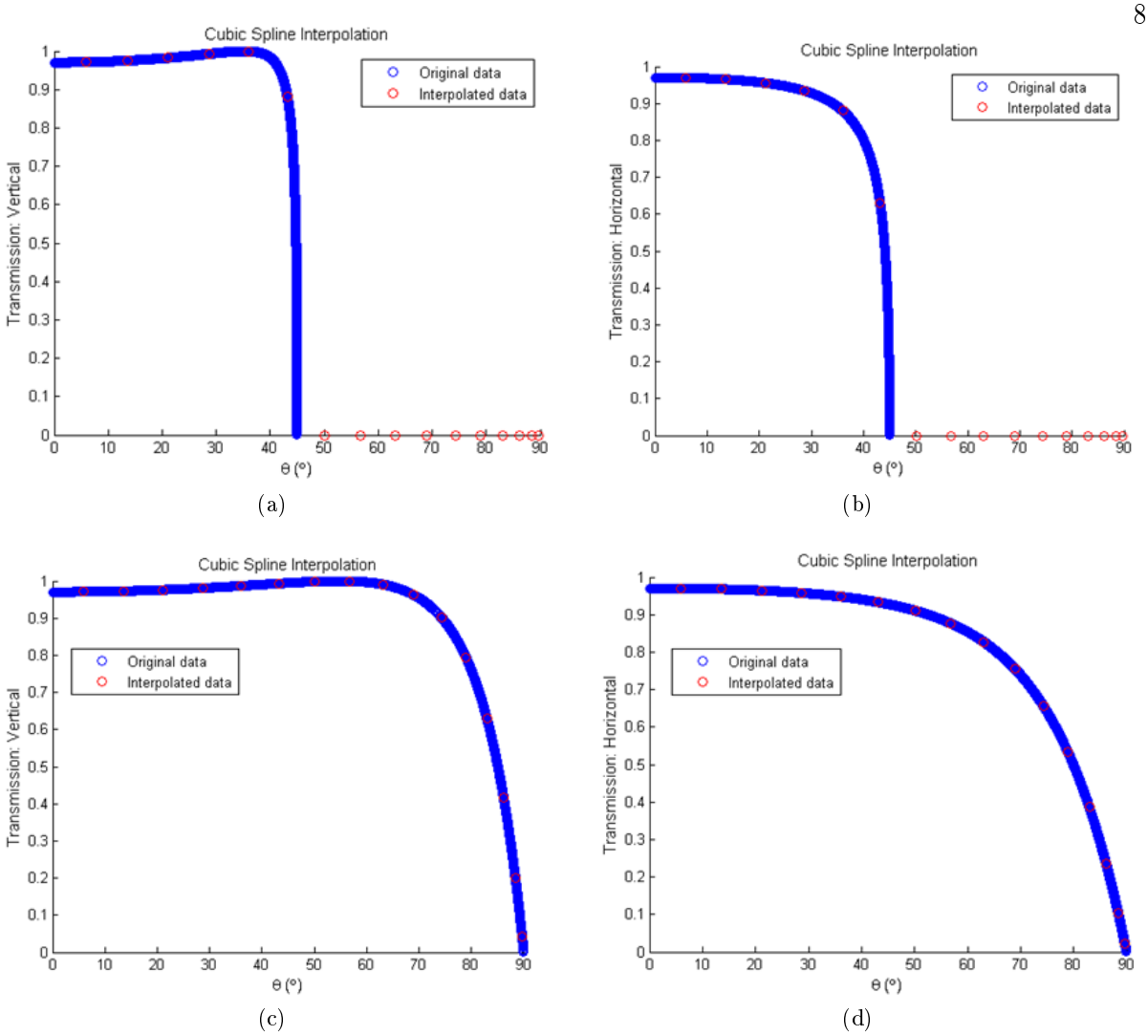


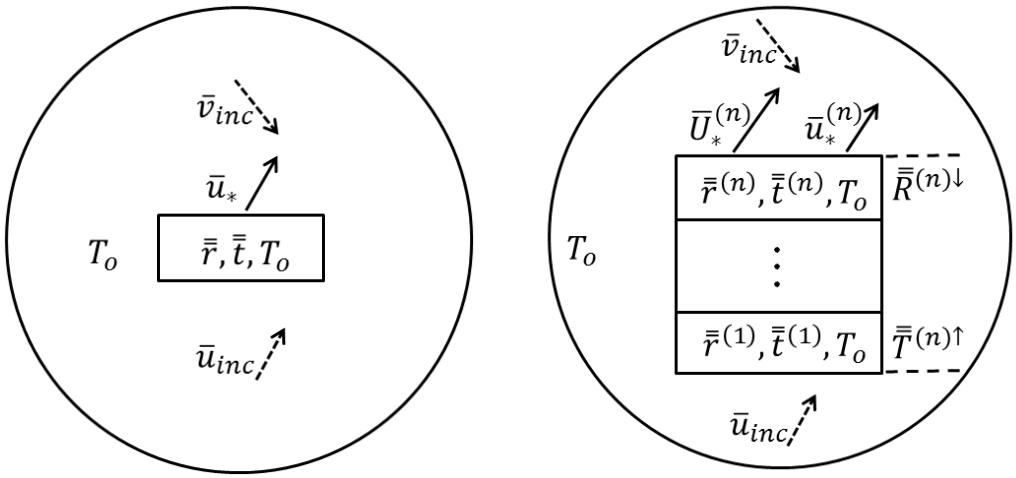
Figure 5.1: Compensation to the refracted streams. (a-b) incident from the layer with $\varepsilon_1 = 1$ to the layer with $\varepsilon_2 = 2$. (c-d) incident from the layer with $\varepsilon_1 = 2$ to the layer with $\varepsilon_2 = 1$.

temperature as the ambient environment, then the stream quantities $\overline{u}_* + \overline{\overline{r}}\,\overline{v}_{inc} + \overline{\overline{t}}\overline{u}_{inc}$ (for a single layer) or $\overline{U}_*^{(n)} + \overline{\overline{R}}^{(n)\downarrow}\overline{v}_{inc} + \overline{\overline{T}}^{(n)\uparrow}\overline{u}_{inc}$ (for a multilayer stack) when scaled to be converted to brightness temperatures must equal the ambient temperature T_o, as illustrated in Fig. 5.2. In this manner UMRT was numerically validated under the conditions of using 1) four different phase matrices (Henyey-Greenstein, Rayleigh, full Mie, and DMRT), 2) two media classes, including rain consisting of a Marshall-Palmer distribution of tenuous liquid water droplets and dry snow consisting of a dense aggregation of ice spheres, 3) a logarithmic frequency grid from 1 to 1000 GHz with

half-decade sampling, and 4) a multilayer stack with up to 6 refracting layers each with different effective permittivity. The worst case of brightness temperature error (similar to that defined in Chapter 3)

$$\varepsilon_{\max}(\theta_i) = \left| \frac{\left(\overline{u}_* + \overline{\overline{r}}\overline{v}_{inc} + \overline{\overline{t}}\overline{u}_{inc}\right)_i}{\sqrt{\mu_i \gamma_i}} - T_o \right| \text{ or } \left| \frac{\left(\overline{U}_*^{(n)} + \overline{\overline{R}}^{(n)\downarrow}\overline{v}_{inc} + \overline{\overline{T}}^{(n)\uparrow}\overline{u}_{inc}\right)_i}{\sqrt{\mu_i \gamma_i}} - T_o \right| \tag{5.1}$$

was found to be $\sim 10^{-10}$ K, indicating excellent stability and accuracy.



(a) (b)

Figure 5.2: Validation of the UMRT solution by imposing energy conservation: (a) for a single layer; (b) for a multilayer stack.

The UMRT Jacobian was validated by comparing the model's analytical derivatives with corresponding numerical counterparts obtained by central divided differencing. Jacobian accuracy is assessed by computing the maximum absolute relative difference between the analytical and numerical derivatives for variations of the relevant input parameters:

$$\varepsilon_{p_n}^{\max} = \lim_{\triangle p_n \to 0} \left| \frac{\frac{\partial T_B(\cdots, p_n, \cdots)}{\partial p_n} - \frac{T_B(\cdots, p_n + \triangle p_n, \cdots) - T_B(\cdots, p_n, \cdots)}{\triangle p_n}}{\frac{\partial T_B(\cdots, p_n, \cdots)}{\partial p_n}} \right| \tag{5.2}$$

where the parameter p_n is an arbitrary radiative parameter (currently, either κ_a, κ_s, T_o, γ, and d) for the n^{th} layer. Examples of $\varepsilon_{p_n}^{\max}$ are provided in Tabs. 5.1 and 5.2 for a single layer and a

multilayer stack, respectively. In Tab. 5.1, $\varepsilon_{p_n}^{\max}$ is provided for four key derivatives under conditions of four relevant phase matrices for a single layer. The thermal radiation at a frequency $f = 10$ GHz for a 1 km thick, 10 mm/hr Marshall Palmer rain layer is simulated using the HG, Rayleigh and Mie phase matrices and linear temperature profile varying from 270 to 280 K. For the case of the DMRT phase matrix a 1 m thick dry snow layer with a 25% snow volume fraction of mean diameter $\langle D \rangle = 0.06$ cm and linear temperature profile varying from 260 to 270 K is assumed. All maximum errors in Tab. 5.1 are much smaller than a percent, indicating a high degree of accuracy for the UMRT Jacobians.

Table 5.1: Assessment of UMRT Jacobian accuracy $\varepsilon_{p_n}^{\max}$ for a single layer.

Numerical Step	1.00E-05			
Medium	Rain			Dry Snow
Phase Matrix	HG	Rayleigh	Mie	DMRT
$\partial T_B/\partial \kappa_s$	3.80E-05	3.36E-05	5.10E-03	1.66E-04
$\partial T_B/\partial \kappa_a$	1.05E-05	1.05E-05	5.76E-05	9.18E-06
$\partial T_B/\partial T_o$	1.18E-09	1.46E-09	1.59E-09	7.44E-10
$\partial T_B/\partial \gamma$	2.90E-08	3.77E-08	4.33E-08	4.28E-08
$\partial T_B/\partial d$	8.73E-04	1.18E-03	1.18E-03	8.74E-04

Derivative errors were similarly studied for a multilayer stack at several observation levels and (without loss of generality) for only the HG phase matrix. The observation levels are depicted in Fig. 5.3 and the error results are provided in Tab. 5.2. In this case, layer 2 (from the bottom) is the Jacobian layer and the observation level is stepped from layer 2 to layer 5. As for the case of this 6-layer stack the values of $\varepsilon_{p_n}^{\max}$ are all also much smaller than a percent. In Tab. 5.2, $T_B^{\uparrow\downarrow}$ denotes the up- and down-welling brightness temperature, respectively.

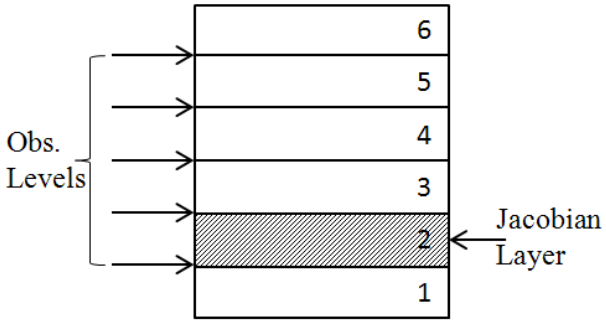


Figure 5.3: Validation of the UMRT Jacobian for a 6-layer stack with the 2nd layer being the Jacobian layer.

Table 5.2: Assessment of UMRT Jacobian accuracy $\varepsilon_{p_n}^{\max}$ for a 6-layer stack with HG phase matrix.

	Perturbation at Layer 2			
Observation at	Layer 2	Layer 3	Layer 4	Layer 5
$\frac{\partial T_B^{\uparrow}}{\partial \kappa_s}$	1.15E-03	1.14E-04	1.32E-04	8.30E-05
$\frac{\partial T_B^{\uparrow}}{\partial \kappa_g}$	4.05E-07	3.98E-07	3.98E-07	3.91E-07
$\frac{\partial T_B^{\uparrow}}{\partial T_g}$	4.93E-11	1.25E-08	1.07E-08	1.19E-09
$\frac{\partial T_B^{\uparrow}}{\partial \gamma}$	1.41E-06	1.40E-06	3.88E-06	3.76E-06
$\frac{\partial T_B^{\uparrow}}{\partial d}$	3.67E-06	5.44E-07	5.43E-07	4.87E-07
Observation at	Layer 5	Layer 4	Layer 3	Layer 2
$\frac{\partial T_B^{\downarrow}}{\partial \kappa_s}$	1.15E-03	3.04E-04	2.95E-04	2.60E-04
$\frac{\partial T_B^{\downarrow}}{\partial \kappa_g}$	4.05E-07	4.21E-07	4.23E-07	4.39E-07
$\frac{\partial T_B^{\downarrow}}{\partial T_g}$	4.93E-11	1.86E-10	1.19E-10	1.28E-10
$\frac{\partial T_B^{\downarrow}}{\partial \gamma}$	4.67E-06	4.66E-06	4.65E-06	4.64E-06
$\frac{\partial T_B^{\downarrow}}{\partial d}$	3.89E-06	7.73E-06	9.75E-07	1.67E-06

5.2 Comparison with Models and Field Data Intercomparison

The upwelling radiation streams obtained using UMRT compare favorably with field measurements over snowpack published by Onstott **et al**. [56]. In this study the upwelling radiation was measured from an 8 cm thick dry snow layer with 11% volumetric ice inclusion and $\langle D \rangle_{ice} = 0.025$ cm over a 5 cm thick nearly-solid ice layer with 1% volumetric air bubble inclusions and $\langle D \rangle_{air} = 0.065$

cm. The frequency was 18.7 GHz and the ambient temperature was 262 K. Based on this information their measurements were simulated using a 4-layer model whose top and bottom layers are the atmosphere ($\varepsilon = 1$) and a homogenous soil background ($\varepsilon = 6.5$), respectively (Fig. 5.4). As shown in the figure the permittivities of the ice ($n = 1$) and dry snow ($n = 2$) layers are computed using DMRT by solving for the effective propagation constant using the the Lorentz-Lorenz law and Ewald-Oseen theorem [37]. A non-trivial part of this calculation is that DMRT uses vacuum ($\varepsilon = 1$) as the background by default and other materials as inclusions, for example, the snow layer in our model. However, for the ice layer the background medium is ice and inclusions are air bubbles ($\varepsilon = 1$). Therefore, the corresponding size parameter and complex refractive index used in the Mie calculation needs to be computed appropriate to these conditions [57] prior to the DMRT calculation. Lacking atmospheric information from [56] the downwelling brightness temperature (V_{top}) was assigned to various values between 0 and 80 K to simulate radiation from the atmosphere.

Layer 3: Atmosphere with $T_{atm}(z)$ and $\varepsilon_3 = 1$
Layer 2: Dry snow (11%) Thickness: 8 cm and $\varepsilon_2 = 1.147$
Layer 1: Ice (1% air) Thickness: 5.2 cm and $\varepsilon_1 = 3.160$
Layer 0: Background with 262 K and $\varepsilon_0 = 6.5$

Figure 5.4: A four-layer UMRT model used by UMRT to simulate the measurements from Onstott **et al**.

The comparison of observed and simulated brightness temperatures is shown in Fig. 5.5. Good agreement is seen between the simulation results and measurements for all angles in the vertical polarization. Changes in the horizontally polarized brightness temperature caused by increasing the atmospheric downwelling radiation are much greater than for vertical polarization, which is a clear

result of the Brewster effect. Importantly, it is noted that the UMRT results successfully capture the upward trend in the horizontally polarized brightness temperature measurements at incidence angles beyond $\sim 60^{o}$. Such trends at steep incidence angles have been reported by several other investigators [58, 59], but to date have not been corroborated by radiative transfer models.

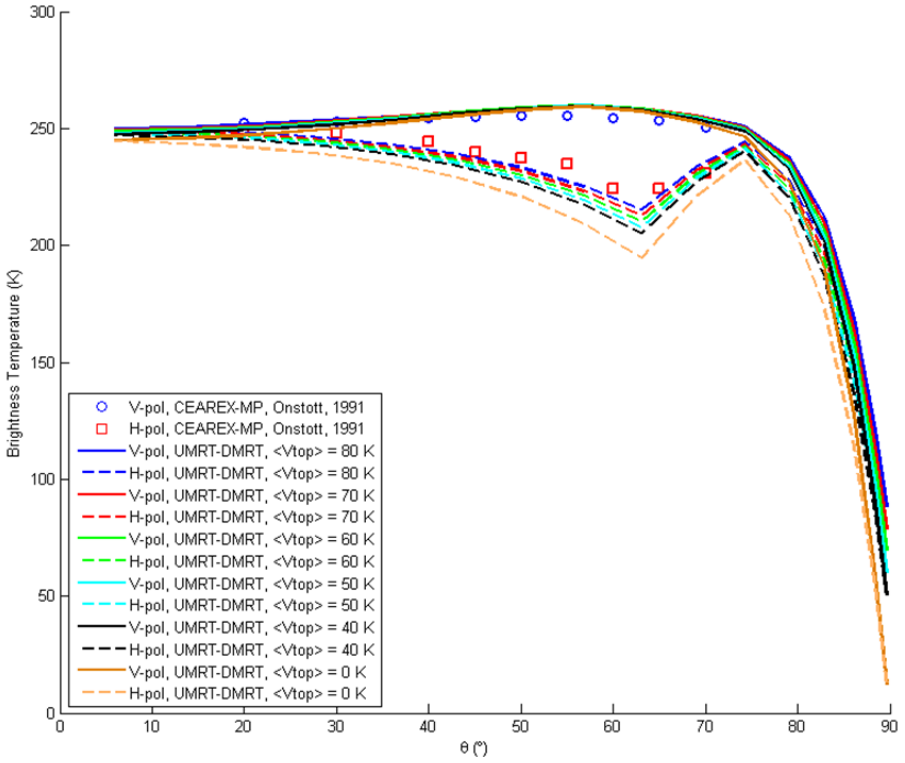


Figure 5.5: Comparison between upwelling radiation streams obtained using the UMRT model and field measurements over snowpack by Onstott **et al**.

To explore the cause of this upward trend the detailed radiation streams in both UMRT and DOTLRT have been studied. Figs. 5.6(a,c,e) show the upwelling self-radiation streams from the ice ($n = 1$) and snow ($n = 2$) layers, along with the radiation from the soil background layer, respectively. Unlike UMRT, DOTLRT assumes layers without refracting boundaries, therefore the background upwelling radiation remains constant over all incident angles (Fig. 5.6(e)). However, UMRT successfully captures the relevant stream refraction. Further, Figs. 5.6(b,d) show the upwelling radiation streams $\overline{U}_{*}^{(1)}$ and $\overline{U}_{*}^{(2)}$ from the ice layer and beneath and from the snow layer and

beneath, respectively. Comparing these two sub-figures, we can see that DOTLRT fails to capture the upward trend, suggesting that boundary refraction is important, and particularly so at steep angles of incidence.

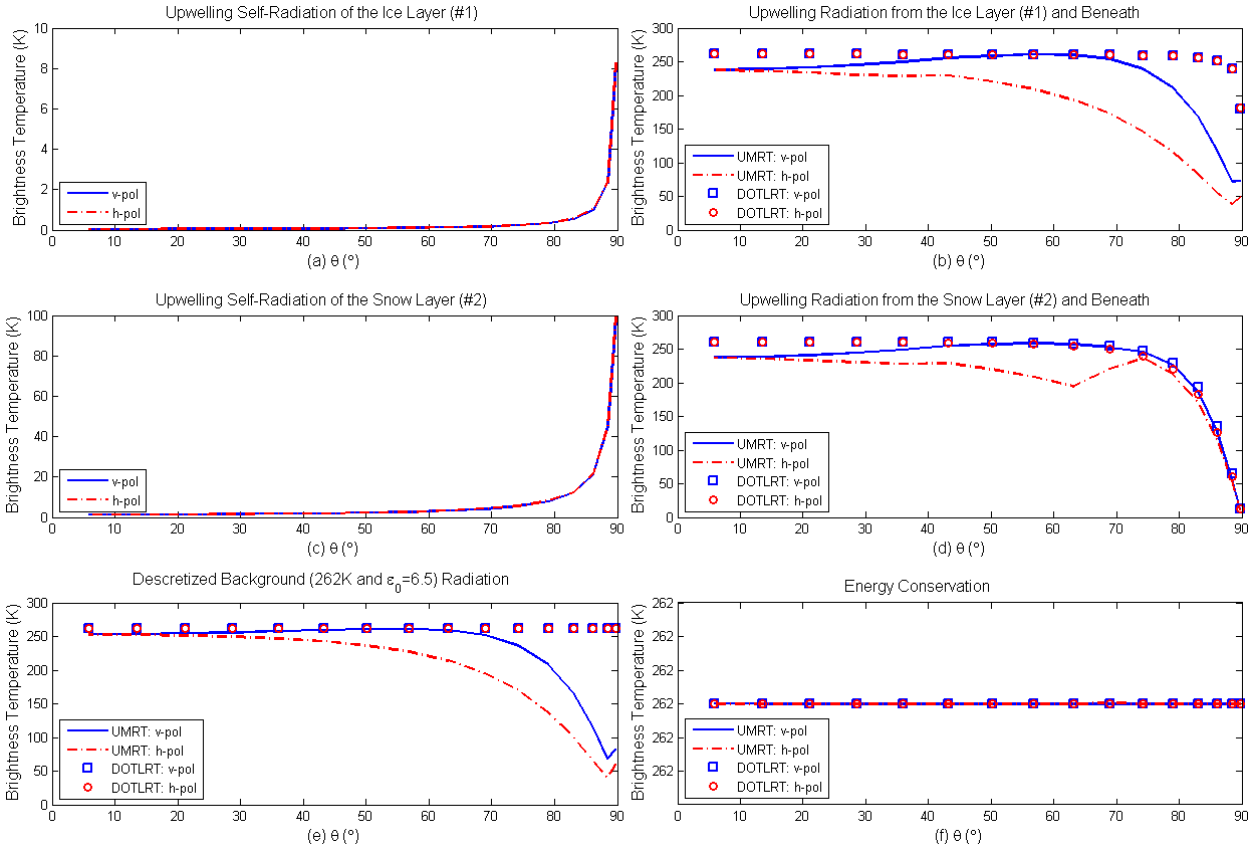


Figure 5.6: Details of the radiation streams for the 4-layer model using both UMRT and DOTLRT.

The importance of the UMRT streams in Fig. 5.6(d) is further explored in Figs. 5.7(a-d). It is seen that the upward trend of Fig. 5.5 is the result of major contributions from $\overline{U}_*^{(1)}$ due to the multiple scattering process modeled in Chapter 4 involving self-radiation streams $\overline{u}_*^{(2)}$ and $\overline{v}_*^{(2)}$.

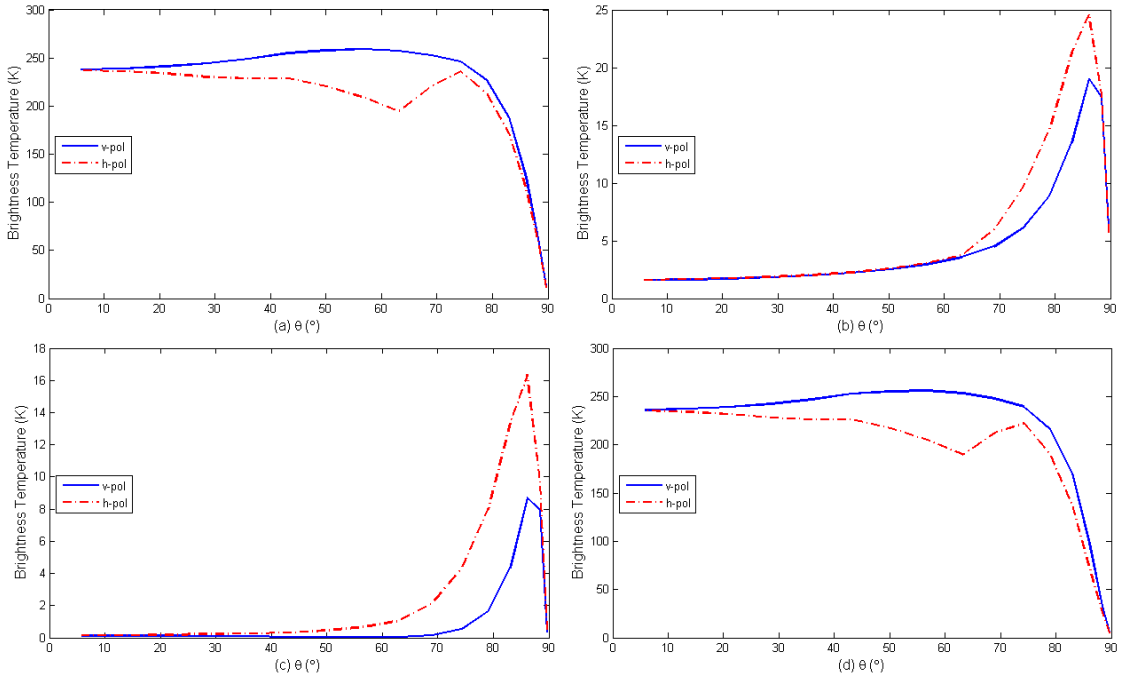


Figure 5.7: Details of the total upwelling radiation $\overline{U}_*^{(2)}$, in (a), along with contributions to $\overline{U}_*^{(2)}$ from three sources: (b-c) the up-welling $\overline{u}_*^{(2)}$ and down-welling $\overline{v}_*^{(2)}$ self-radiation streams from the snow layer (#2), and (d) the upwelling radiation streams $\overline{U}_*^{(1)}$ from the stack underneath the snow layer, including the ice layer and background.

Finally, in Fig. 5.6(f), validation of energy conservation for both UMRT and DOTLRT is illustrated by assuming the downwelling radiation from the upper half space is $V_{top} = 262$ K.

The second comparison uses the field measurements over a snowpack published by Schanda and Matzler, 1981 [60] (Fig. 5.8, taken from [60]). In this study, the upwelling radiation was measured from a 1.5 m thick dry snow layer with an average physical temperature 268 K under the winter atmosphere, which is given in the bottom of Fig. 5.8.

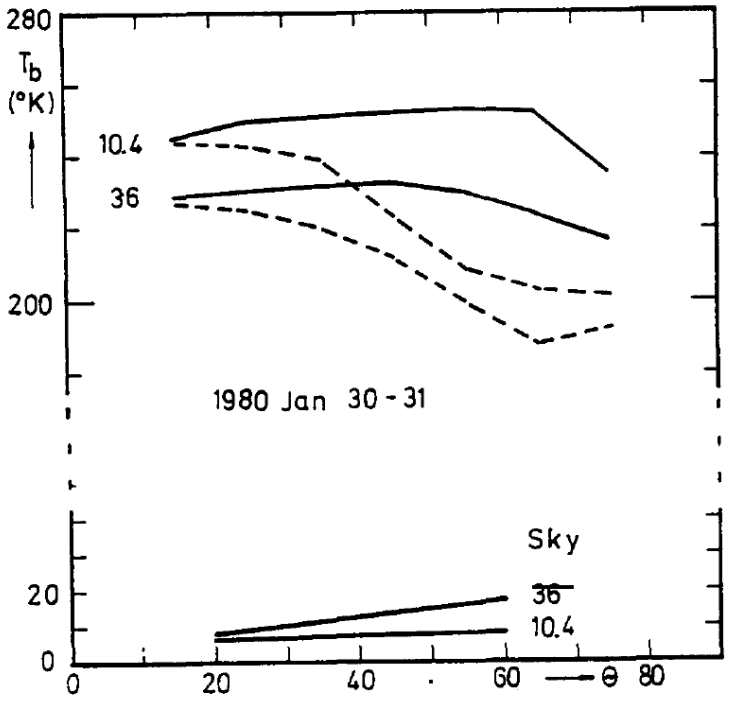
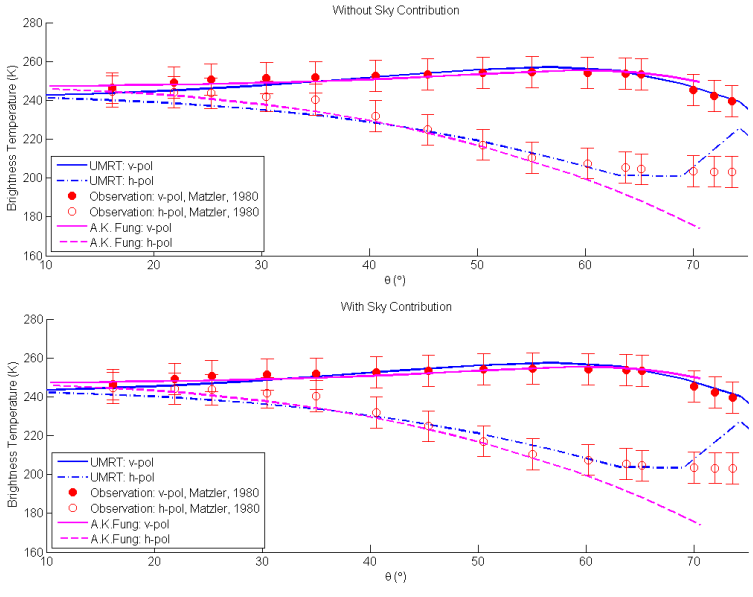


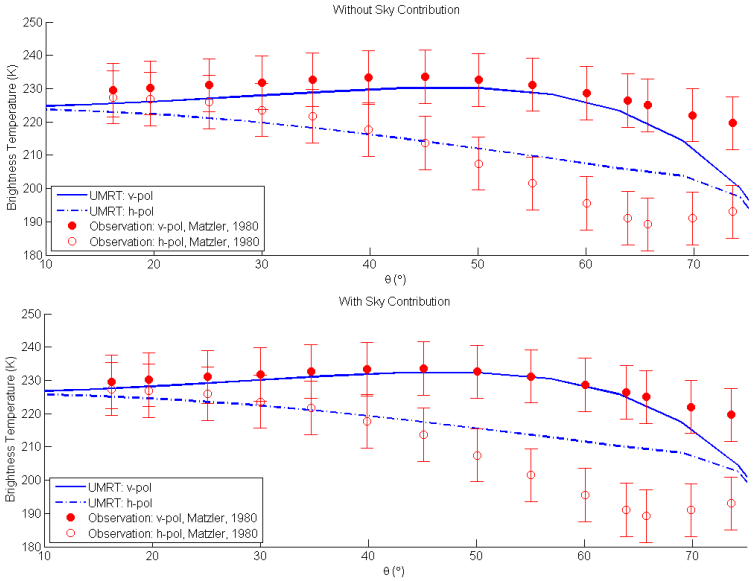
Figure 5.8: Upwelling brightness temperatures of a snowpack measured under the sky at winter condition at 10.4 and 36.0 GHz. Solid and dashed lines: vertical and horizontal polarization, respectively.

Fig. 5.9(a) shows the comparison between the observed [60] and simulated brightness temperatures from both UMRT and Fung's model [42] at 10.4 GHz. Similar to the first example, good agreement is seen between both the simulation results and measurements for all angles in the vertical polarization, although clearly, the UMRT successfully captures the upward trend in the horizontally polarized brightness temperature measurements. For the case of 36.0 GHz as seen in Fig. 5.9(b), the UMRT results agree well with the measurements for all angles less than $\sim 70^{o}$ in the vertical polarization but show a greater discrepancy in the horizontal polarization, especially at large angles beyond 50^{o} (and reaching an error of $\sim 5\%$ at $\sim 55^{o}$).

The third comparison uses field measurements collected during the NASA Cold Land Process Experiment (CLPX) 2003 by the University of Tokyo's Ground-Based Microwave Radiometer-7 (GBMR-7) system at 18.7 and 36.5 GHz, with incidence angles ranging from 30 to 70^{o}. Figs.



(a)

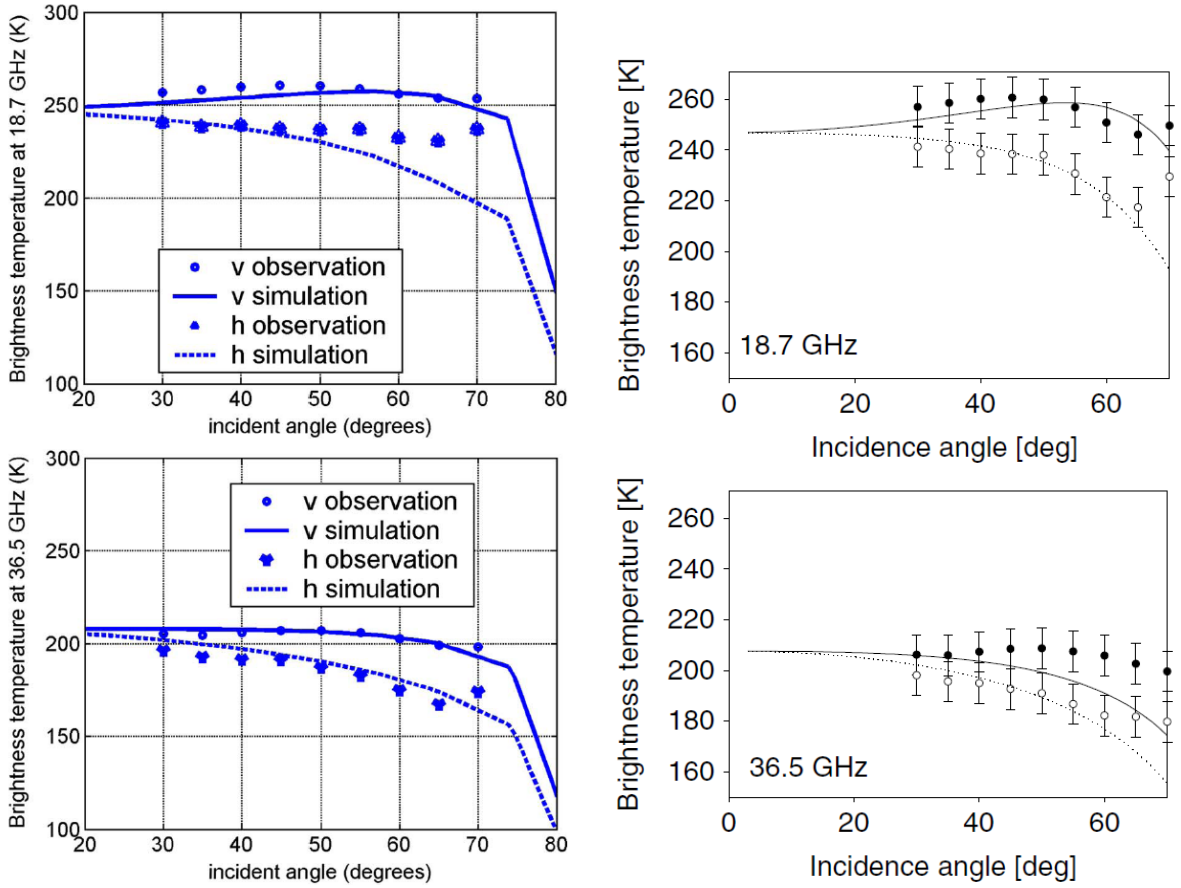


(b)

Figure 5.9: Comparison of upwelling radiation streams obtained using the UMRT model, Fung's model and field measurements over snowpack at: (a) 10.4 GHz and (b) 36.0 GHz. The error bars on measured brightness temperatures represent an error of ±8 K, accounting for a variation of ±2 mm on the particle size.

5.10(a-b) show the simulation results from Liang **et al.**, 2008 [59] and Tedesco **et al.**, 2006 [61] based on the data measured at the same snow pit but on two different days, February 20 and February 21, respectively. According to their publications, both Liang's and Tedesco's models are

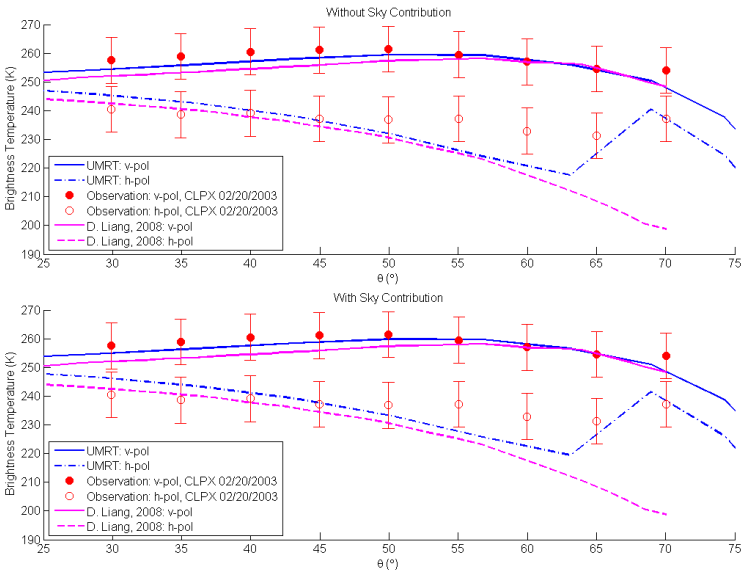
multilayer and based on DMRT-QCA theory, but Liang's model includes the effects of multiple layers of dry snow on microwave scattering, reflection and emission, and thus yields a slightly better results, seen in Figs. 5.10(a-b). Therefore, in this study Liang's model is used for intercomparison with UMRT.



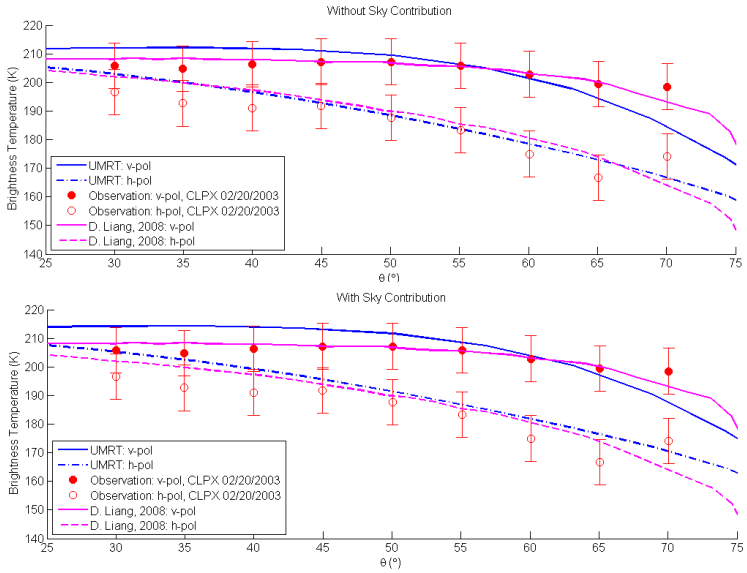
(a) (b)

Figure 5.10: Comparison of the simulated upwelling brightness temperatures with the CLPX 2003 GBMR-7 measurements at 18.7 and 36.5 GHz: (a) Liang **et al.**, 2008 compared with February 20, 2003. (b) Tedesco **et al.**, 2006 compared with February 21, 2003.

In this comparison, the UMRT model was run for both 18.7 and 36.5 GHz through a detailed dry snow parameter space consisting of equal sampling in particle mean diameter $\langle D \rangle$ from 0.02 to 0.24 cm (12 values), volume fraction f_v from 10 to 40% (7 values), and 11 samples in layer thickness d from 0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1, and 1.5 m. The examples shown in Figs. 5.11(a-b) are calculated using a three-layer UMRT model, in which the bottom and top layers are



(a)



(b)

Figure 5.11: Comparison of the simulated upwelling brightness temperatures between UMRT and Liang's model along with the CLPX 2003 GBMR-7 measurements at: (a) 18.7 and (b) 36.5 GHz. The downwelling brightness temperature of atmosphere in both simulations are assumed to be 10 K as reference.

the homogeneous background ($\varepsilon = 3.2$) and atmosphere ($\varepsilon = 1.0$), respectively, and the middle layer is a 0.5 m dry snow layer with $\langle D \rangle = 0.04$ cm and $f_v = 15\%$. Although we note that in this study, there are quite a few other parameter points that can reproduce predicted values of the upwelling

radiation, it is seen in Fig. 5.11(a) that the UMRT model predicts better in both polarizations than that of Liang's model at 18.7 GHz. Specifically, there is significant improvement in horizontal polarization at large observation angles. For 36.5 GHz, the UMRT model predicts slightly better than that of Liang's model for horizontal polarization at large observation angles where it has a ~5 K difference (slightly worse than Liang's model) in vertical polarization.

The last comparison is a preliminary study of the upwelling radiation from the Teshekpuk coastal plain lake, Alaska measured during the NOAA ARCTIC06 on March 26, 2006.

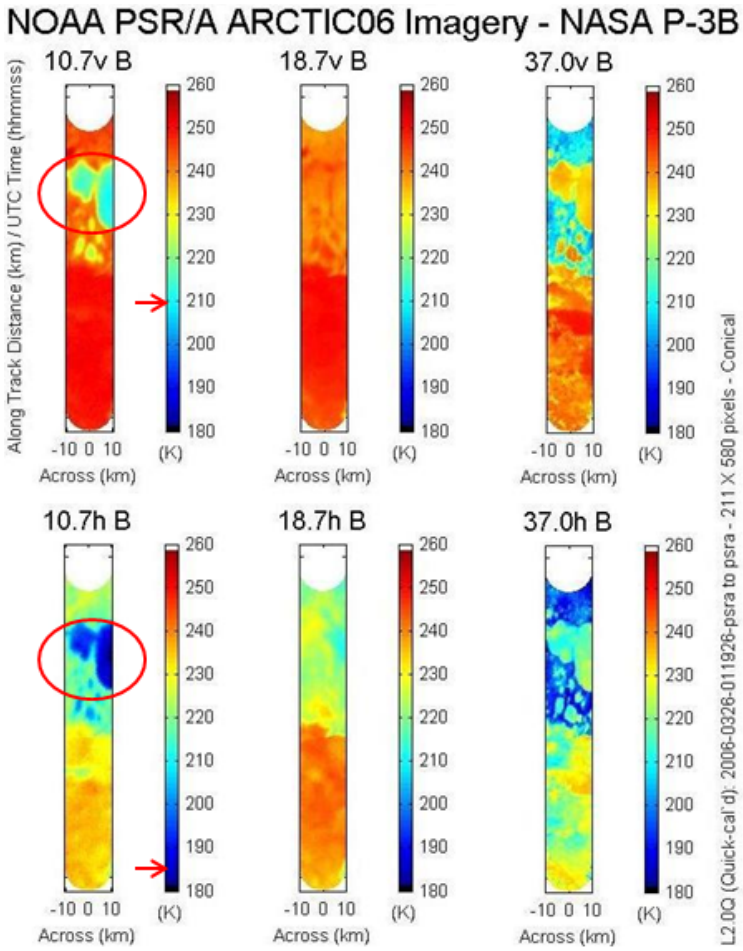


Figure 5.12: NOAA PSR/A ARCTIC06 Imagery - NASA P-3B at 10.7, 18.7 and 37.0 GHz. The circled (red) area is the Teshekpuk lake.

The red circled area in Fig. 5.12 is the Teshekpuk lake and in this figure, the upwelling

brightness temperatures measured at 10.7 GHz are very "cold" (comparing with that measured at the other two frequencies): vertical ~210±5 K and horizontal ~185±5 K. After running the UMRT model through a similar parameter profiling (1260 cases in total): equally sampled in $\langle D \rangle$ from 0.10:0.02:0.20 cm, f_v from 10:5:40 (%), and d from 0.05:0.05:1.5 m, the best estimation we found is: vertical ~217 K and horizontal ~190 K for the case of a half meter dry snow layer with $\langle D \rangle = 0.18$ cm and $f_v = 20\%$.

Chapter 6

Conclusions

This dissertation presents research on the development and validation of a unified microwave radiative transfer (UMRT) model with Jacobian applicability for microwave radiance assimilation, and subsequently, on the upwelling brightness temperature comparisons among UMRT, other radiative transfer models, and field measurements. The UMRT model presented herein is developed for rapid, stable and accurate level-centric calculation of the thermal radiation emitted from any geophysical medium comprised of planar layers of either densely or tenuously distributed spherical scatterers of moderate electrical size. UMRT is derived under the framework of the discrete ordinate tangent linear radiative transfer model (DOTLRT) but combines several unique features from Mie scattering theory and dense media radiative transfer (DMRT) theory .

Important features of UMRT include: 1) the inherent stability and high computational efficiency of recursive matrix calculations performed for obtaining both the brightness temperatures and associated Jacobians, 2) incorporation of full Mie and DMRT phase matrices for both sparse and dense media, 3) exact solutions for the radiation streams under the condition that the temperature profiles within each layer being linearized, 4) accurate compensation of refracted radiation streams by applying a cubic spline interpolation based on the calculation from the Fresnel's and Snell's laws, and 5) extended Jacobian formulation accommodating to the use of a multilayer stack with refracting layers, the DMRT theory and the linear temperature profile. As a result of these extensions UMRT is applicable to real-time all-weather microwave radiance assimilation in both clear and cloudy atmospheres and over both simple and dense volume-scattering media for both

atmospheric and surface nowcasting and forecasting.

The entire UMRT Jacobian formulation has been programmed in Matlab, and validated through both energy conservation and numerical Jacobian intercomparison. As shown in Chapter 5, the model produces upwelling brightness temperatures that agree well with field measurements over dry snow and ice specifically at steep incidence angles.

6.1 Summary of Thesis

In Chapter 2 the microwave radiative transfer theory is briefly reviewed. Simplification of the phase matrix, extinction matrix and absorption vector under the symmetry assumption from using both spherical particle approximation and the planar-stratified approximation, is discussed. To ensure the applicability of the stable matrix operation formulation of DOTLRT, a numerical proof of the symmetry properties of both the Mie and DMRT-QCA phase matrices is given in this chapter.

A summary of the equations for the four phase matrices: Henyey-Greenstein, Rayleigh, Mie and DMRT-QCA and (correspondingly) their associated extinction, scattering and absorption coefficients is included and discussed along with the numerical results and intercomparisons of these phase matrices and coefficients. The significant difference between using the Mie (thus, HG and Rayleigh) theory and the DMRT theory has been clearly shown through the comparisons. Although currently there is conclusive study on the stability of DMRT-QCA algorithm, through our brief study of the DMRT-QCA stability, we conclude that for microwave remote sensing of snow and ice, DMRT-QCA is readily computable for the most practical snow and ice sensing frequencies and particle sizes.

In Chapter 3, the single layer formulation of UMRT for a general planar-stratified structure without refracting layers was derived and discussed. Although the formulation is very similar to that of the DOTLRT, two new features: coupling between vertical and horizontal radiations and linear temperature profile, are introduced into the development of the UMRT single layer solution. Thus, in contrast to DOTLRT, UMRT is a more widely applicable polarimetric (three Stokes' parameters)

and level-centric (rather than layer-centric) discrete-ordinate radiative transfer model.

The numerical results of the upwelling brightness temperatures of the 1-km rain case calculated from using the HG, Rayleigh and Mie phase matrices have shown slight differences on the use of the three types of phase matrix (~2 K at 37.0 GHz) and the vertical and horizontal brightness temperatures (Mie, ~6 K at 37.0 GHz). For the 0.1-m dry snow case (using DMRT phase matrix), the difference between vertical and horizontal brightness temperatures is much significant at high frequency (>~10 K, 89.0 GHz) than at lower frequencies, and the brightness temperatures at normal incidence decrease as frequency increases, which is what expected since the snow layer appears less emissive at higher frequencies.

In Chapter 4, the new refractivity adjusted reflection and transmission matrices were first derived. Then, the extended UMRT Jacobian formulation based on the multilayer structure with refracting layers was developed. In this chapter, the Jacobians were: the scattering and absorption coefficients κ_s and κ_a, the layer temperature T_o, the temperature lapse rate γ, and the layer thickness d.

In Chapter 5, first the entire UMRT Jacobian formulation were validated via both energy conservation and numerical Jacobian intercomparison. Then we compared the upwelling brightness temperatures of three snow/sea ice measurements with the calculations of UMRT model and other theoretical models. The comparisons have shown that UMRT agrees well with the field measurements and has overall better accuracy than the other models.

6.2 Suggestions of Future Research

In review of the development of UMRT, two future extensions naturally follow from the current model.

1. Inclusion of rough surface models and corresponding Jacobian models could be incorporated with the current UMRT. Several rough surface scattering models have been developed during the past three decades, including the small perturbation and phase perturbation methods, Kirchhoff model, numerical Monte Carlo models based on full 3D solutions to Maxwell's equations. Incorpo-

ration of these rough surface models at layer boundaries is an ongoing study which will extend the generality of UMRT.

2. the UMRT Jacobian for the radiation parameters could extended further to provide derivatives with respect to more fundamental physical parameters, such as scatterers mean diameter, atmospheric pressure and water vapor density, soil moisture, cloud water density, and stickiness and volume fraction parameters in DMRT.

Moreover, during the comparisons with field measurements, it would be very useful if a fast DMRT (along with Jacobian) library as a function of frequency, particle size and permittivity, layer thickness, stickiness and volume fraction parameters can be well established.

Bibliography

[1] C. T. Swift and R. E. Mcintosh, "Considerations for microwave remote sensing of ocean-surface salinity," Geoscience and Remote Sensing, IEEE Transactions on, pp. 480–491, Oct. 1983.

[2] F. J. Wentz, "Measurement of oceanic wind vector using satellite microwave radiometers," Geoscience and Remote Sensing, IEEE Transactions on, vol. 30, pp. 960–972, Sept. 1992.

[3] J. R. Piepmeier and A. J. Gasiewski, "High-resolution passive microwave polarimetric mapping of ocean surface wind vector fields," Geoscience and Remote Sensing, IEEE Transactions on, vol. 39, pp. 606–622, Mar. 2001.

[4] T. Meissner and F. J. Wentz, "An updated analysis of the ocean surface wind direction signal in passive microwave brightness temperatures," Geoscience and Remote Sensing, IEEE Transactions on, vol. 40, pp. 1230–1240, June 2002.

[5] A. J. Gasiewski and D. H. Staelin, "Statistical precipitation cell parameter estimation using passive 118-ghz O_2 observations," J. Geophys. Res., vol. 94, no. D15, pp. 18 367–18 378, Dec. 1989.

[6] J. G. Ferriday and S. K. Avery, "Passive microwave remote sensing of rainfall with ssm/i: Algorithm development and implementation," J. Appl. Meterol., vol. 33, pp. 1587–1596, Dec. 1994.

[7] D. H. Staelin and F. W. Chen, "Precipitation observation near 54 and 183 ghz using the noaa 15 satellite," Geoscience and Remote Sensing, IEEE Transactions on, vol. 38, pp. 2322–2332, Sept. 2000.

[8] N. T. Kurtz, T. Markus, D. J. Cavalieri, L. C. Sparling, W. Krabill, A. J. Gasiewski, and J. G. Sonntag, "Estimation of sea ice thickness distributions through the combination of snow depth and satellite laser altimetry data," J. Geophys. Res., Oct. 2009.

[9] F. Weng and N. C. Grody, "Retrieval of cloud liquid water using the special sensor microwave imager (ssm/i)," J. Geophys. Res., vol. 99, no. D12, pp. 25 535–25 551, Dec. 1994.

[10] G. M. Skofronick Jackson and A. J. Gasiewski, "Nonlinear statistical retrievals of ice content and rain rate using passive microwave observations of a simulated convective storm," Geoscience and Remote Sensing, IEEE Transactions on, vol. 33, pp. 957–970, July 1995.

[11] M. Klein and A. J. Gasiewski, "The sensitivity of millimeter and sub-millimeter frequencies to atmospheric temperature and water vapor variations," J. Geophys. Res. - Atmospheres, vol. 13, pp. 17 481–17 511, July 2000.

[12] P. E. Racette, E. R. Westwater, Y. Han, A. J. Gasiewski, M. Klein, D. Cimini, D. C. Jones, W. Manning, E. J. Kim, J. R. Wang, V. Leuski, and P. Kiedron, "Measurements of low amounts of precipitable water vapor using ground-based millimeterwave radiometry," J. Atmos. Ocean. Tech., vol. 22, no. 4, pp. 317–337, April 2005.

[13] R. Bindlish, T. J. Jackson, A. J. Gasiewski, M. Klein, and E. G. Njoku, "Soil moisture mapping and amsr-e validation using the psr in smex02," Rem. Sensing Env., vol. 103, no. 2, pp. 127–139, July 2006.

[14] T. J. Jackson, A. J. Gasiewski, A. Oldak, M. Klein, E. G. Njoku, A. Yevgrafov, S. Christiani, and R. Bindlish, "Soil moisture retrieval using the c-band polarimetric scanning radiometer during the southern great plains 1999 experiment," Geoscience and Remote Sensing, IEEE Transactions on, vol. 40, no. 10, pp. 2151–2161, Oct. 2002.

[15] M. A. Janssen, Atmospheric Remote Sensing by Microwave Radiometry, Chapter 3. Wiley, 1993.

[16] A. J. Gasiewski, "Atmospheric temperature sounding and precipitation cell parameter estimation using passive 118-GHz O_2 observations," Ph.D. dissertation, Massachusetts Institute of Technology, 1988.

[17] L. Tsang, J. A. Kong, and K. H. Ding, Scattering of Electromagnetic Waves, vol. I. Wiley, 2000.

[18] K. Stamnes and H. Dale, "A new look at the discrete ordinate method for radiative transfer calculation in anisotropically scattering atmospheres," J. Atmos. Sci., vol. 38, pp. 387–399, 1981.

[19] T. Nakajima and M. Tanaka, "Matrix formulations for the transfer of solar radiation in a plane-parallel atmosphere," J. Quant. Spectrosc. Rad. Transf., vol. 35, pp. 13–21, 1986.

[20] K. Stamnes, S. C. Tsay, W. Wiscombe, and K. Jayaweera, "Numerically stable algorithm for discrete ordinate method radiative transfer in multiple scattering and emitting layered media," Appl. Opt., vol. 27, pp. 2502–2529, 1988.

[21] L. Tsang, J. A. Kong, K. H. Ding, and C. O. Ao, Scattering of Electromagnetic Waves, vol. II. Wiley, 2000.

[22] A. G. Voronovich, A. J. Gasiewski, and B. L. Weber, "A fast multistream scattering-based jacobian for microwave radiance assimilation," Geoscience and Remote Sensing, IEEE Transactions on, vol. 42, pp. 1749–1761, Aug 2004.

[23] J. S. Marshall and W. M. Palmer, "The distribution of raindrops with size," J. Meteorol., vol. 5, pp. 165–166, Aug. 1948.

[24] J. Joss, J. C. Thams, and A. Waldvogel, "The variation of rain drop size distribution at locarno," Proc. Conf. Cloud Phys., pp. 369–373, 1967.

[25] A. C. Best, "The size distribution of raindrops," Quart. J. Roy. Meteor. Soc., vol. 76, pp. 16–36, 1950.

[26] D. Atlas and C. W. Ulbrich, "The physical basis for attenuation-rainfall relationships and the measurement of rainfall parameters by combined attenuation and rader methods," J. Rech. Atomos., vol. 8, pp. 275–298, 1974.

[27] M. Sekine and G. Lind, "Rain attenuation of centimeter, millimeter and submillimeter radio waves," Proc. of the 12th European Microwave Conf., pp. 584–589, 1982.

[28] E. Villermaux and B. Bossa, "Single-drop fragmentation determines size distribution of raindrops," Nature Physics, vol. 5, pp. 697–702, Sep. 2009.

[29] R. S. Sekhon and R. C. Srivastava, "Snow size spectra and radar reflectivity," J. Atm. Sci., vol. 27, pp. 299–307, Mar. 1970.

[30] K. L. S. Gunn and J. S. Marshall, "The distribution with size of aggregate snowflakes," J. Meteorol., vol. 15, pp. 452–461, Dec. 1957.

[31] S. Kinne and K. Liou, "The effects of the nonsphericity and size distribution of ice crystals on the radiative properties of cirrus clouds," Atmospheric Research,, vol. 24, pp. 273–284, 1989.

[32] G. M. McFarquhar and A. J. Heymsfield, "Parameterization of tropical cirrus ice crystal size distributions and implications for radiative transfer: Results from cepex," J. of Atmospheric Sci., vol. 54, pp. 2187–2200, Sep. 1997.

[33] B. F. Ryan, "A bulk parameterization of the ice particle size distribution and the optical properties in ice clouds," J. of Atmospheric Sci., vol. 57, pp. 1436–1451, May 2000.

[34] L. Tsang, "Dense medium radiative transfer theory comparison with experimentand application to microwave remote sensing and polarimetry," Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, no. 1, pp. 46–59, Jan. 1990.

[35] ——, "Dense media radiative transfer theory for dense discrete random media with particles of multiple sizes and permittivities," Progress in Electromagnetics Research, PIER 06, pp. 181–230, 1992.

[36] L. Tsang and J. A. Kong, Scattering of Electromagnetic Waves, vol. III. Wiley, 2000.

[37] L. Tsang, J. Pan, D. Liang, Z. Li, D. W. Cline, and Y. Tan, "Modeling active microwave remote sensing of snow using dense media radiative transfer (dmrt) theory with multiple-scattering effects," Geoscience and Remote Sensing, IEEE Transactions on, vol. 45, no. 4, pp. 990–1004, Apr. 2007.

[38] S. Chandrasekhar, Radiative Transfer. New York: Dover, 1960.

[39] A. Ishimaru, Wave Propagation and Scattering in Random Media. New York: Academic, 1978.

[40] L. Tsang, J. A. Kong, and R. Shin, Theory of Microwave Remote Sensing. Wiley-Interscience, New York, 1985.

[41] F. T. Ulaby and C. Elachi, Radar Polarimetry for Geoscience Applications. Artech House Inc., 1990.

[42] A. K. Fung, Microwave Scattering and Emission Models and their Applications, Chapter 8. Artech House, 1994.

[43] C. Matzler, Thermal Microwave Radiation Applications for Remote Sensing. London: IET, 2006.

[44] M. Sekine, S. Ishii, and S. Sayama, "Weibull raindrop-size distribution and its application to rain attenuation from 30 GHz to 1000 GHz," Int. J. Infrared Milli. Waves, pp. 383–392, March 2007.

[45] C. F. Bohren and D. R. Huffman, Absorption and Scattering of Light by Small Particles. Wiley, 1998.

[46] H. C. Van de Hulst, Light Scattering by Small Particles. Dover, 1957.

[47] L. Tsang, C. Chen, A. T. Chang, J. Guo, and K. H. Ding, "Dense media radiative transfer theory based on quasicrystalline approximation with applications to passive microwave remote sensing of snow," Radio Science, vol. 35, no. 3, pp. 731–749, May 2000.

[48] S. G. Warren, "Optical constants of ice from the ultraviolet to the microwave," Applied Optics, vol. 23, pp. 1206–1225, 1983.

[49] K. K. Tse, L. Tsang, C. H. Chan, K. H. Ding, and K. W. Leung, "Multiple scattering of waves by dense random distributions of sticky particles for applications in microwave scattering by terrestrial snow," Radio Science, vol. 42, pp. 1–14, Sep. 2007.

[50] T. Meissner and F. J. Wentz, "The complex dielectric constant of pure and sea water from microwave satellite observations," Geoscience and Remote Sensing, IEEE Transactions on, vol. 42, no. 9, pp. 1836–1849, Sep. 2004.

[51] S. Wolf and N. V. Voshchinnikov, "Mie scattering by ensembles of particles with very large size parameters," Computer Physics Communications, pp. 113–123, 2004.

[52] Y. Takano and K. N. Liou, "Phase matrix for light scattering by concentrically stratified spheres: Comparison of geometric optics and the exact theory," Applied Optics, vol. 49, no. 20, pp. 3990–3996, July 2010.

[53] Y. Choi and J. Cheong, "New expressions of 2x2 block matrix inversion and their application," Automatic Control, IEEE Transactions on, vol. 54, no. 11, pp. 2648–2653, Nov. 2009.

[54] M. Tian and A. J. Gasiewski, "A unified microwave radiative transfer model for general planar stratified media: Slab formulation," In Preparation, 2012.

[55] K. Atkinson, An Introduction to Numerical Analysis, 2nd ed. Wiley, 1989.

[56] R. Onstott, "Active microwave observations of arctic sea ice during the fall freeze-up," Proc. IGARSS 91 Symp, June 1991.

[57] F. T. Ulaby, R. K. Moore, and A. K. Fung, Microwave Remote Sensing: Active and Passive, vol. I, Chapter 5. Artech House Inc., 1981, vol. 1.

[58] C. Matzler, E. Schanda, R. Hofer, and W. Good, "Microwave signatures of the natural snow cover at weissfluhjoch," NASA Conference Publication 2153, pp. 203–223, 1980.

[59] D. Liang, X. Xu, and L. Tsang, "The effects of layers in dry snow on its passive microwave emissions using dense media radiative transfer theory based on the quasicrystalline approximation (qca/dmrt)," Geoscience and Remote Sensing, IEEE Transactions on, vol. 46, no. 11, pp. 3663–3671, Nov 2008.

[60] E. Schanda and C. Matzler, "Optimum characteristics for snow pack evaluation by microwave radiometry," Adv. Space Res., vol. 1, pp. 151–162, 1981.

[61] M. Tedesco, E. Kim, D. Cline, T. Graf, T. Koike, R. Armstrong, M. Brodzik, and J. Hardy, "Comparison of local scale measured and modelled brightness temperatures and snow parameters from the CLPX 2003 by means of a dense medium radiative transfer theory model.pdf," Hydrol. Process., vol. 20, pp. 657–672, 2006.

[62] L. A. Klein and C. T. Swift, "An improved model for the dielectric constant of sea water at microwave frequencies," Antenna and Propagation, IEEE Transactions on, vol. AP-25, no. 1, pp. 104–111, Jan. 1977.

[63] T. Meissner and F. J. Wentz, "The complex dielectric constant of pure and sea water from microwave satellite observations," Geoscience and Remote Sensing, IEEE Transactions on, vol. 42, no. 9, pp. 1836–1849, Sep. 2004.

[64]

[65]

[66]

[67] F. J. Wentz and T. Meissner, "AMSR ocean algorithm (version 2)," Remote Sens. Syst., Santa Rosa, CA, 1999.

[68] H. J. Liebe, G. A. Hufford, and T. Manabe, "A model for the complex permittivity of water at frequencies below 1 THz," Int. J. Infr. Millim. Waves, vol. 12, no. 7, pp. 659–675, 1991.

[69] C. Guillou and et al., "Impact of new permittivity measurements on sea surface emissivity modeling in microwaves," Radio Sci., vol. 33, no. 3, pp. 649–667, May 1998.

[70] J. R. Wang, "A comparison of the MIR-estimated and model-calculated fresh water surface emissivities at 89, 150, and 220 GHz," Geoscience and Remote Sensing, IEEE Transactions on, vol. 40, pp. 1356–1365, June 2002.

[71] A. Stogryn and et al., "The microwave permittivity of sea and fresh water," Azusa, CA: GenCorp Aerojet, 1995.

[72] J. e. a. Barthel, "A computer-controlled system of transmission lines for the determination of the complex permittivity of lossy liquids between 8.5 and 90 GHz," Ber. Bunsenges. Phys. Chem., vol. 95, no. 8, pp. 853–859, 1991.

[73] D. Bertolini, M. Cassettari, and G. Salvetti, "The dielectric relaxation time of supercooled water," J. Chem. Phys., vol. 76, no. 6, pp. 3285–3290, 1982.

[74] U. Kaatze and V. Uhlendorf, "The dielectric properties of water at microwave frequencies," Zeitsch f Phys. Chem. Neue Folge, vol. 126, pp. 151–165, 1981.

[75] J. B. Hasted and et al., "The temperature variation of the near millimeter wavelength optical constants of water," Infr. Phys., vol. 27, no. 1, pp. 11–15, 1987.

[76] M. E. Tiuri, A. H. Sihvola, E. G. Nyfors, and M. T. Hallikaiken, "The complex dielectric constant of snow at microwave frequencies," Oceanic Engi. IEEE Journal of, vol. OE-9, no. 5, pp. 377–382, Dec. 1984.

[77] T. Achammer and A. Denoth, "Snow dielectric properties from dc to microwave x-band," Annals of Glaciology, vol. 19, pp. 92–96, 1994.

[78] A. Denoth, "An electronic device for long-term snow wetness recording," Annals of Glaciology, vol. 19, pp. 104–106, 1994.

[79] A. Kovacs, A. Gow, and R. M. Morey, "A reassessment of the in-situ dielectric constant of polar film," CREEL Rep. 93-26, 1993.

[80] A. Wiesmann and C. Matzler, "Microwave emission model of layered snowpacks," Remote Sensing of Environment, vol. 70, no. 3, pp. 307–316, Dec. 1999.

[81] H. Looyenga, "Dielectric constants of heterogeneous mixture," Physica, vol. 31, pp. 401–406, 1965.

[82] C. Matzler, "Microwave permittivity of dry snow," Geoscience and Remote Sensing, IEEE Transactions on, vol. 34, no. 2, pp. 573–581, March 1996.

[83] A. Sihvola and J. A. Kong, "Effective permittivity of dielectric mixtures," Geoscience and Remote Sensing, IEEE Transactions on, vol. 26, pp. 420–429, 1988.

[84] C. F. Bohren and L. J. Battan, "Radar backscattering of microwave by spongy ice spheres," J. Atmosph. Sci., vol. 39, pp. 2623–2628, 1982.

[85] D. Polder and J. H. Van Santen, "The effective permeability of mixtures of solids," Physica., vol. 12, no. 5, pp. 257–271, 1946.

[86] C. Matzler and A. Wiesmann, "Extension of the microwave emission model of layered snowpacks to coarse-grained snow," Remote Sensing of Environment, vol. 70, no. 3, pp. 317–325, Dec. 1999.

[87] A. D. Frolov and Y. Y. Macheret, "On dielectri properties of dry and wet snow," Hydrological Processes, vol. 13, pp. 1755–1760, March 1999.

[88] A. Denoth, "Snow dielectric measurements," Adv. Space Res., vol. 9, no. 1, pp. 233–243, 1989.

[89] C. Matzler and U. Wegmuller, "Dielectric properties of freshwater ice at microwave frequencies," J. Phys. D: Applied Phys., vol. 20, pp. 1623–1630, April 1987.

[90] P. S. Ray, "Broadband complex refractive indices of ice and water," Appl. Opt., vol. 11, no. 8, pp. 1836–1844, Aug. 1972.

Appendix A

Scattering Amplitude Matrix

The scattering amplitude matrix is derived for both the Rayleigh and Mie scattering by a homogeneous sphere in the particle-based system of coordinates (Fig. A.1, also refereed as the 1-2 system in chapter 1, [17]), where $\hat{k}_i$ and $\hat{k}_s$ are the propagation directions of the complex incident and scattered electric fields $\overline{E}_i$ and $\overline{E}_s$ (respectively) and the directions of $\hat{1}_{i,s}$, $\hat{2}_{i,s}$ and $\hat{k}_{i,s}$ are such that they are orthonormal unit vectors following the right-hand rule. The derivation follows that in Tsang, et al., 2000 [17].

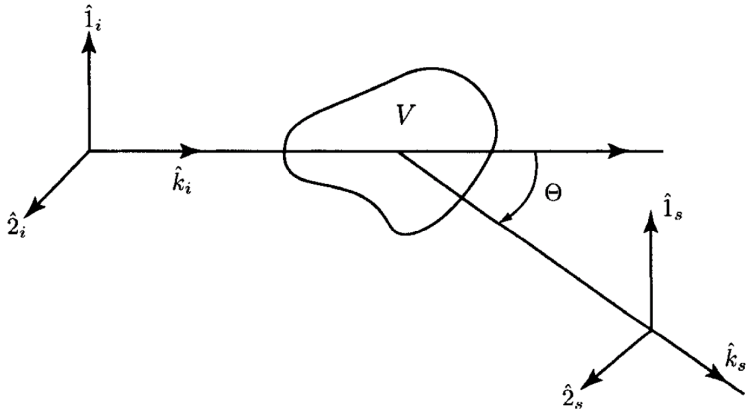


Figure A.1: Geometry of the particle-based coordinate system. The scattering plane contains $\hat{k}_i$ and $\hat{k}_s$. The angle between $\hat{k}_i$ and $\hat{k}_s$ is Θ, called the forward scattering angle.

In the particle-based coordinate system, under the assumption of independent scattering the amplitude and polarization of an incident plane wave scattered by a single particle to any direction at a distance r can be described by

$$\overline{E}_s = \frac{e^{-jkr}}{r}\overline{\overline{F}}\left(\Theta\right)\cdot\overline{E}_i \tag{A.1}$$

where $\overline{\overline{F}}\left(\Theta\right)$ is the scattering amplitude matrix and $k = \frac{2\pi}{\lambda}$ is the wavenumber in air.

Using the 1-2 orthonormal basis, Eq. (A.1) is written as

$$\begin{bmatrix} E_{1s} \\ E_{2s} \end{bmatrix} = \frac{e^{-jkr}}{r}\begin{bmatrix} f_{11}\left(\Theta\right) & f_{12}\left(\Theta\right) \\ f_{21}\left(\Theta\right) & f_{22}\left(\Theta\right) \end{bmatrix}\begin{bmatrix} E_{1i} \\ E_{2i} \end{bmatrix} \tag{A.2}$$

Let

$$\hat{1}_i = \hat{1}_s = \frac{\hat{k}_s \times \hat{k}_i}{\left|\hat{k}_s \times \hat{k}_i\right|} \tag{A.3}$$

By orthonormality,

$$\begin{aligned} \hat{2}_i &= \hat{k}_i \times \hat{1}_i \\ \hat{2}_s &= \hat{k}_s \times \hat{1}_s \end{aligned} \tag{A.4}$$

Other useful relations for this 1-2 system are

$$\begin{aligned} &\hat{k}_s \cdot \hat{k}_i = \cos\Theta \\ &\hat{2}_s \cdot \hat{2}_i = \hat{k}_s \cdot \hat{k}_i = \cos\Theta \\ &\hat{k}_s \cdot \hat{2}_i = \hat{k}_s \cdot \hat{k}_i \times \hat{1}_i = \hat{k}_s \times \hat{k}_i \cdot \hat{1}_i = \sin\Theta \end{aligned} \tag{A.5}$$

A.1 Rayleigh Scattering

A.1.1 Rayleigh Scattering by a Small Particle

In Rayleigh scattering, the particle diameter is much less than wavelength, $D \ll \lambda$. In this case of electrically small particle a dipole moment $\overline{p}$ is induced inside the particle. The far field radiated by the dipole moment $\overline{p}$ in the direction $\hat{k}_s$ is

$$\overline{E}_s = -\frac{k^2 e^{-jkr}}{4\pi\epsilon r}\hat{k}_s \times \left(\hat{k}_s \times \overline{p}\right) \tag{A.6}$$

Let the electric field inside the particle be denoted by $\overline{E}_{int}$, where the subscript **int** denotes internal. The polarization per unit volume inside the particle is

$$\overline{P}_{int} = (\epsilon_p - \epsilon)\,\overline{E}_{int} \tag{A.7}$$

where ϵ_p is the permittivity of the particle.

In Rayleigh scattering, the internal field of the particle is a constant vector. The dipole moment of the particle is

$$\overline{p} = v_o \overline{P}_{int} \tag{A.8}$$

where v_o is the volume of the particle.

Substituting Eqs. (A.7) and (A.8) into Eq. (A.6) yields

$$\begin{aligned} \overline{E}_s = & -\tfrac{k^2 e^{-jkr}}{4\pi\epsilon r}\hat{k}_s \times \left[\hat{k}_s \times v_o\,(\epsilon_p - \epsilon)\,\overline{E}_{int}\right] \\ = & \tfrac{k^2 e^{-jkr}}{4\pi\epsilon r} v_o\,(\epsilon_p - \epsilon)\left[\hat{1}_s\left(\hat{1}_s \cdot \overline{E}_{int}\right) + \hat{2}_s\left(\hat{2}_s \cdot \overline{E}_{int}\right)\right] \end{aligned} \tag{A.9}$$

Via Eq. (A.9), elements of the scattering amplitude matrix for the Rayleigh scattering by a small particle can be determined by relating the internal field $\overline{E}_{int}$ to the incident field $\overline{E}_i$. The scattering amplitude matrix for the case of the particle being sphere is derived next.

A.1.2 Rayleigh Scattering by a Sphere

Considering a sphere of radius $a \ll \lambda$ centered at the origin, the internal field of the particle is

$$\overline{E}_{int} = \frac{3\epsilon}{\epsilon_p + 2\epsilon}\overline{E}_i \tag{A.10}$$

and the internal field $\overline{E}_{int}$ is parallel to the incident field $\overline{E}_i$. Substituting Eq. (A.10) into Eq. (A.9) and note that the volume of a sphere of radius a is $v_o = \frac{4\pi a^3}{3}$,

$$\overline{E}_s = \frac{e^{-jkr}}{r} f_o\left[\hat{1}_s\left(\hat{1}_s \cdot \overline{E}_i\right) + \hat{2}_s\left(\hat{2}_s \cdot \overline{E}_i\right)\right] \tag{A.11}$$

where

$$f_o = k^2 a^3 \frac{\epsilon_p - \epsilon}{\epsilon_p + 2\epsilon} \tag{A.12}$$

From Eq. (A.11)

$$E_{1s} = f_o \left(\hat{1}_s \cdot \overline{E}_i \right) \tag{A.13}$$

$$E_{2s} = f_o \left(\hat{2}_s \cdot \overline{E}_i \right) \tag{A.14}$$

In this case, the scattering amplitude matrix can be explicitly determined by using the 1-2 orthonormal system. First let $E_{1i} = 1$ and $E_{2i} = 0$, thus $\overline{E}_i = \hat{1}_i$. Substituting these quantities into Eqs. (A.13) and (A.14) and using the orthonormal relations yields

$$\begin{aligned} E_{1s}(\Theta) &= f_{11}(\Theta) = f_o \left(\hat{1}_s \cdot \overline{E}_i \right) = f_o \left(\hat{1}_s \cdot \hat{1}_i \right) = f_o \\ E_{2s}(\Theta) &= f_{21}(\Theta) = f_o \left(\hat{2}_s \cdot \overline{E}_i \right) = f_o \left(\hat{2}_s \cdot \hat{1}_i \right) = 0 \end{aligned} \tag{A.15}$$

Next let $E_{1i} = 0$ and $E_{2i} = 1$, thus $\overline{E}_i = \hat{2}_i$. Similarly, substituting these quantities into Eqs. (A.13) and (A.14) and using the unit vector relations in Eq. (A.5) yields

$$\begin{aligned} E_{1s}(\Theta) &= f_{12}(\Theta) = f_o \left(\hat{1}_s \cdot \overline{E}_i \right) = f_o \left(\hat{1}_s \cdot \hat{2}_i \right) = 0 \\ E_{2s}(\Theta) &= f_{22}(\Theta) = f_o \left(\hat{2}_s \cdot \overline{E}_i \right) = f_o \left(\hat{2}_s \cdot \hat{2}_i \right) = f_o \cos\Theta \end{aligned} \tag{A.16}$$

Thus the scattering amplitude matrix for the Rayleigh scattering by a sphere has the following simple form:

$$\begin{bmatrix} f_{11}(\Theta) & f_{12}(\Theta) \\ f_{21}(\Theta) & f_{22}(\Theta) \end{bmatrix} = \begin{bmatrix} f_o & 0 \\ 0 & f_o \cos(\Theta) \end{bmatrix} \tag{A.17}$$

Note that if the Eqs. (A.13) and (A.14) are expressed in the principal coordinate system, which is defined by $(\theta_s, \phi_s; \theta_i, \phi_i)$, then by applying the coordinates transformation described in Chapter 2 of this thesis, the scattering amplitudes are

$$
\begin{aligned}
f_{11}(\theta_s, \phi_s; \theta_i, \phi_i) &= f_o \left[\cos\theta_s \cos\theta_i \cos(\phi_s - \phi_i) + \sin\theta_s \sin\theta_i\right] \\
f_{21}(\theta_s, \phi_s; \theta_i, \phi_i) &= -f_o \cos\theta_i \sin(\phi_s - \phi_i) \\
f_{12}(\theta_s, \phi_s; \theta_i, \phi_i) &= f_o \cos\theta_s \sin(\phi_s - \phi_i) \\
f_{22}(\theta_s, \phi_s; \theta_i, \phi_i) &= f_o \cos(\phi_s - \phi_i)
\end{aligned}
\tag{A.18}
$$

A.2 Mie Scattering by Sphere

The scattering amplitude matrix for the Mie scattering by a sphere of radius a and permittivity ϵ_p centered at the origin (Fig. A.2, taken from Fig. 1.6.1 in [17]) is derived as follows.

To get $\begin{bmatrix} f_{11}(\Theta) & f_{12}(\Theta) \\ f_{21}(\Theta) & f_{22}(\Theta) \end{bmatrix}$, let the incident wave be along the $\hat{z}$ direction, so that $\hat{k}_i = \hat{z}$. Also let the observation direction $\hat{k}_s$ be in the $y - z$ plane, with $\phi = 90^o$; then

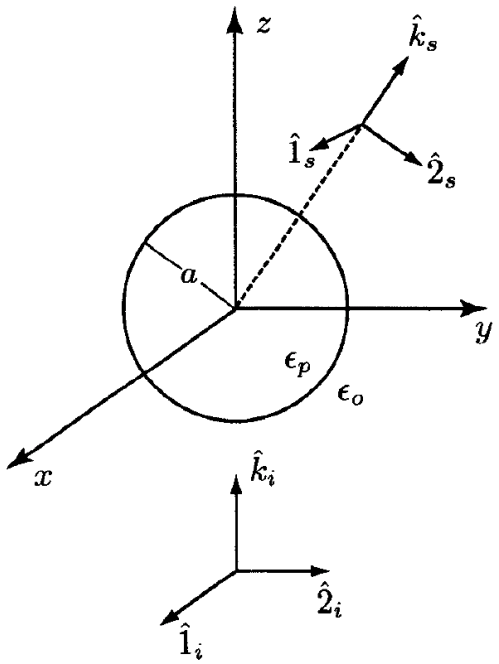


Figure A.2: Geometry of a sphere with radius a and permittivity ϵ_p. The incident wave propagates in the $+\hat{z}$ direction.

$$\begin{aligned}
\Theta &= \theta \\
\hat{1}_i &= \hat{1}_s = \hat{x} = -\hat{\phi} \\
\hat{2}_i &= \hat{y} \\
\hat{2}_s &= \hat{k}_s \times \hat{1}_s = \hat{\theta}
\end{aligned} \tag{A.19}$$

Skip some intermediate steps,

$$\begin{aligned}
\overline{E}_s = \quad & \frac{je^{-jkr}}{2kr}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\{-a_n\left[\hat{e}_i\cdot(\hat{x}+j\hat{y})\right]\left[\hat{\theta}\tau_n(\cos\Theta)-\hat{\phi}j\pi_n(\cos\Theta)\right]e^{-j\phi} \\
& -a_n\left[\hat{e}_i\cdot(\hat{x}-j\hat{y})\right]\left[\hat{\theta}\tau_n(\cos\Theta)+\hat{\phi}j\pi_n(\cos\Theta)\right]e^{j\phi} \\
& -jb_n\left[\hat{e}_i\cdot(\hat{x}+j\hat{y})\right]\left[-\hat{\theta}j\pi_n(\cos\Theta)-\hat{\phi}\tau_n(\cos\Theta)\right]e^{-j\phi} \\
& -jb_n\left[\hat{e}_i\cdot(\hat{x}-j\hat{y})\right]\left[-\hat{\theta}j\pi_n(\cos\Theta)+\hat{\phi}\tau_n(\cos\Theta)\right]e^{j\phi}\}
\end{aligned} \tag{A.20}$$

where (a_n, b_n) are the Mie scattering coefficients,(π_n, τ_n) are the angle-dependent functions and $\hat{e}_i$ is the propagation direction of the incident electric field.

To get $f_{11}(\Theta)$ and $f_{21}(\Theta)$, we follow the relations in Eq. (A.19) and first set $\hat{e}_i = \hat{x}$, $\phi = 90^o$, so that $\hat{1}_i = \hat{1}_s = \hat{x} = -\hat{\phi}$ and $\hat{2}_s = \hat{\theta}$. Substituting these quantities into Eq. (A.20) yields

$$-a_n\left[\hat{e}_i\cdot(\hat{x}+j\hat{y})\right]\left[\hat{\theta}\tau_n(\cos\Theta)-\hat{\phi}j\pi_n(\cos\Theta)\right]e^{-j\phi} = ja_n\left[\hat{\theta}\tau_n(\cos\Theta)-\hat{\phi}j\pi_n(\cos\Theta)\right]$$

$$-a_n\left[\hat{e}_i\cdot(\hat{x}-j\hat{y})\right]\left[\hat{\theta}\tau_n(\cos\Theta)+\hat{\phi}j\pi_n(\cos\Theta)\right]e^{j\phi} = -ja_n\left[\hat{\theta}\tau_n(\cos\Theta)+\hat{\phi}j\pi_n(\cos\Theta)\right]$$

$$-jb_n\left[\hat{e}_i\cdot(\hat{x}+j\hat{y})\right]\left[-\hat{\theta}j\pi_n(\cos\Theta)-\hat{\phi}\tau_n(\cos\Theta)\right]e^{-j\phi} = -b_n\left[-\hat{\theta}j\pi_n(\cos\Theta)-\hat{\phi}\tau_n(\cos\Theta)\right]$$

$$-jb_n\left[\hat{e}_i\cdot(\hat{x}-j\hat{y})\right]\left[-\hat{\theta}j\pi_n(\cos\Theta)+\hat{\phi}\tau_n(\cos\Theta)\right]e^{j\phi} = b_n\left[-\hat{\theta}j\pi_n(\cos\Theta)+\hat{\phi}\tau_n(\cos\Theta)\right]$$

and thus

$$\begin{aligned} \overline{E}_s = & \frac{e^{-jkr}}{r}\frac{j}{2k}\hat{\phi}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\left\{2\left[a_n\pi_n(\cos\Theta)+b_n\tau_n(\cos\Theta)\right]\right\} \\ = & \frac{e^{-jkr}}{r}\frac{j}{k}\hat{\phi}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\left[a_n\pi_n(\cos\Theta)+b_n\tau_n(\cos\Theta)\right] \end{aligned} \tag{A.21}$$

Since $\hat{\phi} = -\hat{1}_i = -\hat{1}_s$ and $\hat{\theta} = \hat{2}_s$, from (A.21)

$$\begin{aligned} f_{11}(\Theta) = & \frac{-j}{k}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\left[a_n\pi_n(\cos\Theta)+b_n\tau_n(\cos\Theta)\right] \\ f_{21}(\Theta) = & \ 0 \end{aligned} \tag{A.22}$$

Similarly, set $\hat{e}_i = \hat{y}$, $\phi = 90^{\circ}$, so that $\hat{2}_i = \hat{y}$, $\hat{1}_s = -\hat{\phi}$, and $\hat{2}_s = \hat{\theta}$. This case Eq. (A.20) yields

$$\overline{E}_s = \frac{e^{-jkr}}{r}\frac{-j}{k}\hat{\theta}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\left[a_n\tau_n(\cos\Theta)+b_n\pi_n(\cos\Theta)\right] \tag{A.23}$$

Thus

$$\begin{aligned} f_{12}(\Theta) = & \ 0 \\ f_{22}(\Theta) = & \frac{-j}{k}\sum_{n=1}^{\infty}\frac{(2n+1)}{n(n+1)}\left[a_n\tau_n(\cos\Theta)+b_n\pi_n(\cos\Theta)\right] \end{aligned} \tag{A.24}$$

Hence, the Mie scattering amplitude matrix for sphere is

$$\begin{bmatrix} f_{11}(\Theta) & f_{12}(\Theta) \\ f_{21}(\Theta) & f_{22}(\Theta) \end{bmatrix} = \begin{bmatrix} f_{11}(\Theta) & 0 \\ 0 & f_{22}(\Theta) \end{bmatrix} \tag{A.25}$$

It can be concluded that the scattering amplitudes $f_{12}(\Theta)$ and $f_{21}(\Theta)$ are zero for both the Rayleigh and Mie scattering by a sphere.

A.3 Connection to Stokes Matrix

Based on the above conclusion, for scattering by a sphere, Eq. (A.1) is written as

$$\begin{bmatrix} E_{1s}(\Theta) \\ E_{2s}(\Theta) \end{bmatrix} = \frac{e^{-jkr}}{r} \begin{bmatrix} f_{11}(\Theta) & 0 \\ 0 & f_{22}(\Theta) \end{bmatrix} \begin{bmatrix} E_{1i}(\Theta) \\ E_{2i}(\Theta) \end{bmatrix} \tag{A.26}$$

, for which the relation between the incident and scattered Stokes parameters follows [45]:

$$\begin{bmatrix} I_s \\ Q_s \\ U_s \\ V_s \end{bmatrix} = \frac{1}{r^2} \begin{bmatrix} S_{11} & S_{12} & 0 & 0 \\ S_{12} & S_{11} & 0 & 0 \\ 0 & 0 & S_{33} & -S_{34} \\ 0 & 0 & S_{34} & S_{33} \end{bmatrix} \begin{bmatrix} I_i \\ Q_i \\ U_i \\ V_i \end{bmatrix} \tag{A.27}$$

where

$$\begin{aligned} S_{11} &= \tfrac{1}{2}\left[|f_{11}(\Theta)|^2 + |f_{22}(\Theta)|^2\right], \quad S_{12} = \tfrac{1}{2}\left[|f_{11}(\Theta)|^2 - |f_{22}(\Theta)|^2\right] \\ S_{33} &= \mathrm{Re}\left\{f_{11}(\Theta) f_{22}^*(\Theta)\right\}, \qquad\qquad S_{34} = \mathrm{Im}\left\{f_{11}(\Theta) f_{22}^*(\Theta)\right\} \end{aligned} \tag{A.28}$$

The relation between the Stokes parameters and the modified Stokes parameters is

$$\begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_v \\ I_h \\ U \\ V \end{bmatrix} \tag{A.29}$$

Substituting Eq. (A.29) into Eq. (A.27) yields

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{vs} \\ I_{hs} \\ U_s \\ V_s \end{bmatrix} = \frac{1}{r^2} \begin{bmatrix} S_{11} & S_{12} & 0 & 0 \\ S_{12} & S_{11} & 0 & 0 \\ 0 & 0 & S_{33} & -S_{34} \\ 0 & 0 & S_{34} & S_{33} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{vi} \\ I_{hi} \\ U_i \\ V_i \end{bmatrix} \tag{A.30}$$

$$\Rightarrow \begin{bmatrix} I_{vs} \\ I_{hs} \\ U_s \\ V_s \end{bmatrix} = \frac{1}{r^2} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} S_{11} & S_{12} & 0 & 0 \\ S_{12} & S_{11} & 0 & 0 \\ 0 & 0 & S_{33} & -S_{34} \\ 0 & 0 & S_{34} & S_{33} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{vi} \\ I_{hi} \\ U_i \\ V_i \end{bmatrix} \tag{A.31}$$

$$\Rightarrow \begin{bmatrix} I_{vs} \\ I_{hs} \\ U_s \\ V_s \end{bmatrix} = \frac{1}{r^2} \begin{bmatrix} S_{11}+S_{12} & 0 & 0 & 0 \\ 0 & S_{11}-S_{12} & 0 & 0 \\ 0 & 0 & S_{33} & -S_{34} \\ 0 & 0 & S_{34} & S_{33} \end{bmatrix} \begin{bmatrix} I_{vi} \\ I_{hi} \\ U_i \\ V_i \end{bmatrix} \tag{A.32}$$

Substituting definitions in (A.28) into Eq. (A.32) yields

$$\begin{bmatrix} I_{vs} \\ I_{hs} \\ U_s \\ V_s \end{bmatrix} = \frac{1}{r^2} \begin{bmatrix} |f_{11}(\Theta)|^2 & 0 & 0 & 0 \\ 0 & |f_{22}(\Theta)|^2 & 0 & 0 \\ 0 & 0 & \mathrm{Re}\{f_{11}(\Theta) f_{22}^*(\Theta)\} & -\mathrm{Im}\{f_{11}(\Theta) f_{22}^*(\Theta)\} \\ 0 & 0 & \mathrm{Im}\{f_{11}(\Theta) f_{22}^*(\Theta)\} & \mathrm{Re}\{f_{11}(\Theta) f_{22}^*(\Theta)\} \end{bmatrix} \begin{bmatrix} I_{vi} \\ I_{hi} \\ U_i \\ V_i \end{bmatrix} \tag{A.33}$$

Eq. (A.33) is the definition of the Stokes matrix for spherical particle used in [37] and Chapter 2 of this thesis.

Appendix B

Permittivity Models

Included is a brief introduction to the permittivity models of liquid pure water and seawater, dry and wet snow, and sea ice used within the UMRT algorithm as one of the medium physical parameters.

B.1 Pure Water and Seawater

A succinct review and summary of the permittivity models for pure water and seawater developed during the past five decades (1954-2004) for microwave radiative transfer calculation purposes can be found in [43] edited by C. Matzler. UMRT includes the following two models for computing the complex permittivities of liquid pure water and seawater: 1) Klein-Swift, 1976 [62] and 2) Meissner-Wentz, 2004 [63]. The two models are interpolation functions based on Debye relaxation law for the permittivity of pure water and seawater as functions of frequency (microwave band), temperature and salinity. It should be pointed out that each of the interpolation functions was constructed by respective authors mainly based on their own experimental data and in ignorance of the other data sets measured more or less contemporaneously.

The Klein-Swift model has been mainly used in microwave radiative transfer calculations so far. It is an interpolation function in the form of a single Debye relaxation law, whose parameters are estimated by fitting the laboratory permittivity measurements by Lane and Saxton [64] for water temperature between 0 °C and 40 °C, and Ho **et al**. at frequency 1.43 GHz [65] and 2.653 GHz [66]. The Klein-Swift model employs the most general form of a single Debye relaxation law,

which is given by

$$\epsilon = \epsilon_\infty + \frac{\epsilon_s - \epsilon_\infty}{1 + (j\omega\tau)^{1-\alpha}} - j\frac{\sigma}{\omega\epsilon_o} \tag{B.1}$$

where $\omega = 2\pi f$ is the radian frequency with f in Hz, ϵ_∞ and ϵ_s are, respectively, the static and infinite frequency dielectric constant, ϵ_o is the permittivity of free space (8.854×10^{-12} F/m), τ is the relaxation time in seconds, σ is the ionic conductivity in mhos/m, and α is an empirical parameter (noted as the Cole-Cole spread factor [67]) that describes the distribution of the relaxation times. In Eq. (B.1), $\sigma = 0$ for the case of pure water. Otherwise the model parameters ϵ_s, τ and σ are all functions of temperature (T, in ^oC) and salinity (S, in parts per thousand, ppt) of sea water. The Klein-Swift model is sufficiently accurate for low frequencies but, as it has been shown by various researchers [68, 69, 67, 70], it is getting increasingly inaccurate as frequency increases and the water temperature gets below 0 °C.

More recently, the Meissner-Wentz model fitted the laboratory permittivity measurements of seawater by Stogryn et al., [71] for selected frequencies (7, 10, 18 ,24, 37, 85.5 and 89 GHz), temperatures (-2, +12, +20, and +30°C for each frequency), and salinities (10, 20, 30, 35, and 40 ppt for 37 GHz and below, and 35 ppt for 85.5 and 89 GHz), and the permittivity measurements of pure water by Barthel et al., Kaatze et al., Bertolini et al., and Hasted et al., [72, 73, 74, 75] for a wide range of frequency from 1.7 to 351 GHz and temperature from −21 °C to 40 °C, with a double Debye relaxation law, which employs the following general form:

$$\epsilon = \epsilon_\infty + \frac{\epsilon_s - \epsilon_1}{1 + j\nu/\nu_1} + \frac{\epsilon_1 - \epsilon_\infty}{1 + j\nu/\nu_2} - j\frac{\sigma}{(2\pi\epsilon_o)\,\nu} \tag{B.2}$$

where ϵ_1 is the intermediate frequency permittivity, ν_1 and ν_2 are the first and second Debye relaxation frequencies (in GHz), respectively, the ionic conductivity σ is calculated by

$$\sigma\,(T,S) = \sigma\,(T,S=35)\cdot P\,(S)\cdot Q\,(T,S) \tag{B.3}$$

where

$$
\begin{aligned}
\sigma\left(T, S=35\right) &= 2.903602 + 8.607 \cdot 10^{-2} T + 4.738817 \cdot 10^{-4} T^2 \\
&\quad -2.991 \cdot 10^{-6} T^3 + 4.3041 \cdot 10^{-9} T^4 \\
P\left(S\right) &= S \tfrac{37.5109 + 5.45216 S + 0.014409 S^2}{1004.75 + 182.283 + S^2} \\
Q\left(T, S\right) &= 1 + \tfrac{\alpha_0 (T-15)}{T + \alpha_1} \\
\alpha_0 &= \tfrac{6.9431 + 3.2841 S - 0.099486 S^2}{84.85 + 69.024 S + S^2} \\
\alpha_1 &= 49.843 - 0.2276 S + 0.00198 S^2
\end{aligned}
$$

and

$$
\begin{aligned}
\epsilon_s\left(T, S\right) = &\quad \epsilon_s\left(T, S=0\right) \exp\left(b_0 S + b_1 S^2 + b_2 T\, S\right) \\
\nu_1\left(T, S\right) = &\quad \nu_1\left(T, S=0\right) \left[1 + S\left(b_3 + b_4 T + b_5 T^2\right)\right] \\
\epsilon_1\left(T, S\right) = &\quad \epsilon_1\left(T, S=0\right) \exp\left(b_6 S + b_7 S^2 + b_8 T\, S\right) \\
\nu_2\left(T, S\right) = &\quad \nu_2\left(T, S=0\right) \left[1 + S\left(b_9 + b_{10} T\right)\right] \\
\epsilon_\infty\left(T, S\right) = &\quad \epsilon_\infty\left(T, S=0\right) \left[1 + S\left(b_{11} + b_{12} T\right)\right]
\end{aligned}
$$

and

$$
\begin{aligned}
\epsilon_s\left(T, S=0\right) = &\quad \tfrac{37088.6 - 82.168 T}{421.854 + T} \\
\epsilon_1\left(T, S=0\right) = &\quad a_0 + a_1 T + a_2 T^2 \\
\nu_1\left(T, S=0\right) = &\quad \tfrac{45 + T}{a_3 + a_4 T + a_5 T^2} \quad \text{(GHz)} \\
\epsilon_\infty\left(T, S=0\right) = &\quad a_6 + a_7 T \\
\nu_2\left(T, S=0\right) = &\quad \tfrac{45 + T}{a_8 + a_9 T + a_{10} T^2} \quad \text{(GHz)}
\end{aligned}
$$

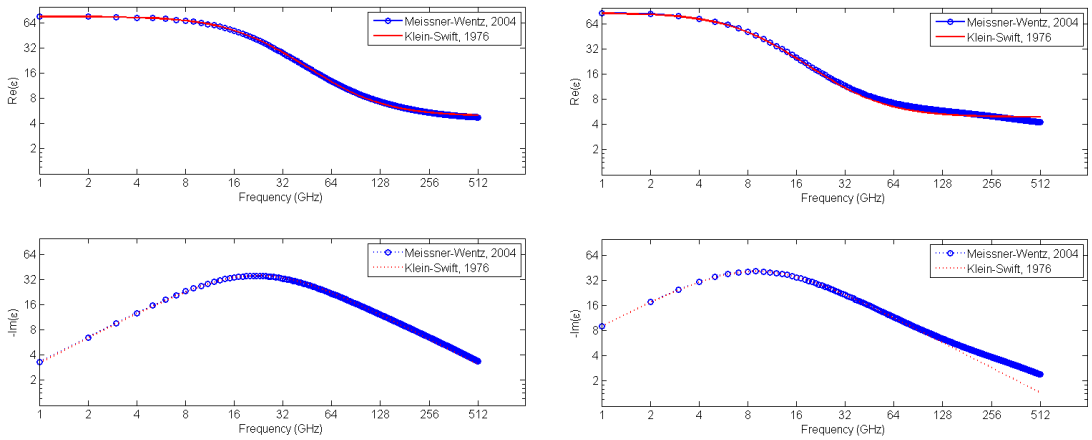
where the coefficients a_i and b_i are given in Tab. I.

Table B.1: Coefficients a_i and b_i used in the Meissner-Wentz (2004) interpolation model

i	a_i	b_i
0	+5.723	−0.00356417
1	+0.022379	+0.00000474868
2	−0.00071237	+0.0000115574
3	+5.0478	+0.00239357
4	−0.070315	−0.000031353
5	+0.00060059	+0.000000252477
6	+3.6143	−0.00628908
7	+0.028841	+0.000176032
8	+0.13652	−0.0000922144
9	+0.0014825	−0.0199723
10	+0.00024166	+0.000181176
11		−0.00204265
12		+0.000157883

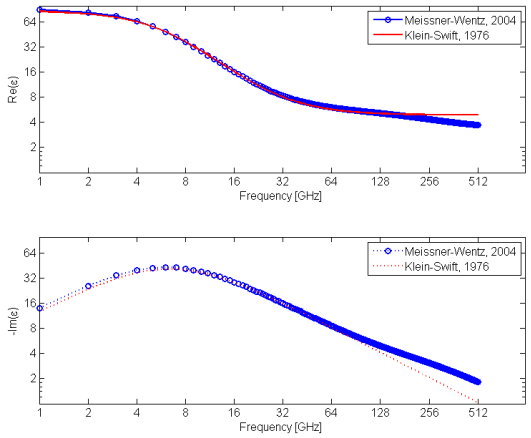
The above double Debye expression derived by Meissner and Wentz are reported has overall better accuracy and wider applicable frequency range (up to 500 GHz) for the permittivity of both pure water and seawater compared with the Klein-Swift model. Another important feature is that the Meissner-Wentz model can be applied to calculate the permittivity of liquid water with a temperature as low as $\tilde{}-42\,^{o}$C. Such cold liquid water is refereed to "supercooled" water, which normally exists when the water is at a pressure down to its crystal homogeneous nucleation. In atmosphere microwave observations presence of the supercooled water should be considered.

Comparisons between the Klein-Swift and Meissner-Wentz models for liquid pure water and seawater are shown in Figs. B.1 and B.2, respectively.

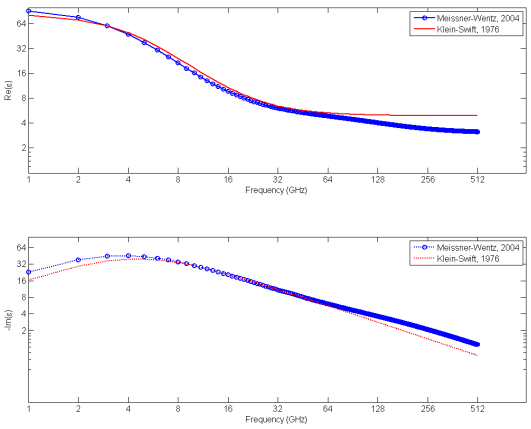


(a) T = 30 °C

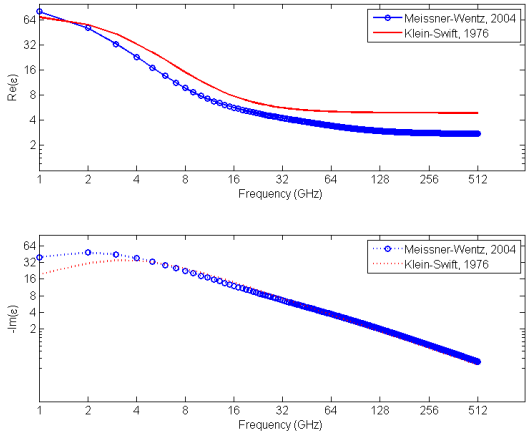
(b) T = 0 °C



(c) T = −10 °C

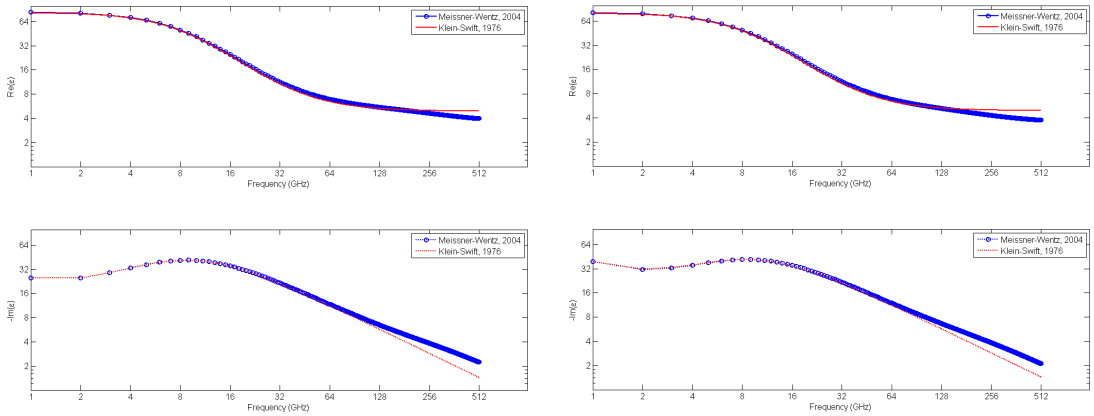


(d) T = −20 °C



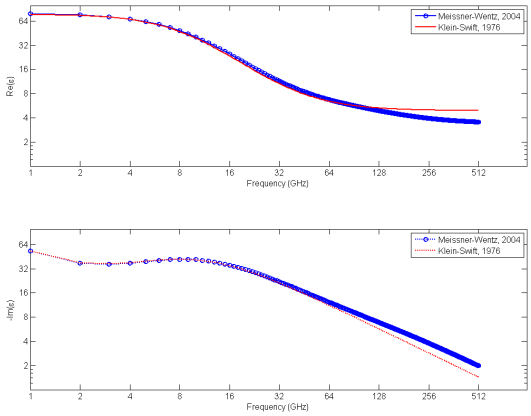
(e) T = −30 °C

Figure B.1: Complex permittivity of pure water calculated using the Klein-Swift and Wentz-Meissner (2004) models for frequency stepped up to 512 GHz and selected temperatures at (a-e) 30, 0, -10, -20, 30 °C, respectively.

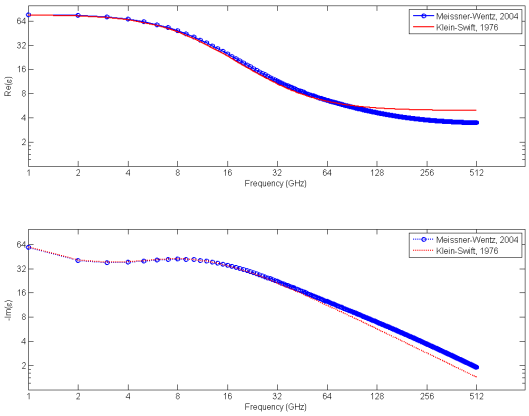


(a) S = 10 ppt

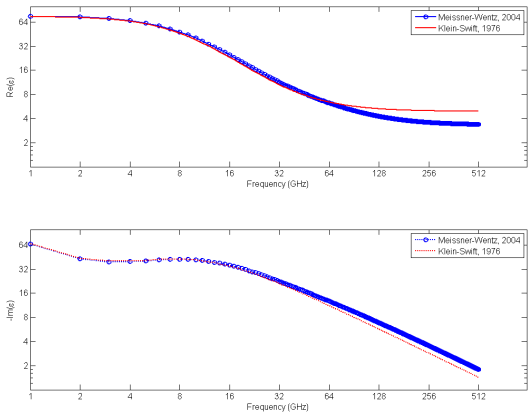
(b) S = 20 ppt



(c) S = 30 ppt



(d) S = 35 ppt



(e) S = 40 ppt

Figure B.2: Complex permittivity of sea water calculated using the Klein-Swift and Wentz-Meissner (2004) models for frequency stepped up to 512 GHz, temperature at 0 °C and selected salinity at (a-e) 10, 20, 30, 35, and 40 ppt, respectively.

In Fig. B.1, it is seen that the two models produce similar permittivity values for pure water at temperatures above 0 °C (subplots a-b) while as expected the difference increases as the water temperature decreases from -10 to -30 °C (subplots c-e). For seawater, it is seen (Fig. B.2) that the two models match well at low frequencies while the discrepancy in both the real and imaginary parts is greater at the high frequency end.

B.2 Dry and Wet Snow

B.2.1 Permittivity of Dry Snow

The real part of the permittivity of dry snow $\epsilon_d^{'}$ can be calculated by the following interpolation functions: 1) Tiuri et al., 1984 [76], 2) Achammer and Denoth, 1994 [77], 3) Denoth, 1994 [78], 4) Kovacs et al., 1993 [79], 5) Wisemann, 1999 [80], 6) Looyenga, 1965 [81], 7) Matzler, 2006 [43], and 8) Matzler, 1996 [82]:

$$
\begin{aligned}
1)\quad \epsilon_d^{'} &= 1 + 1.7\rho_d + 0.7\rho_d^2 \\
2)\quad \epsilon_d^{'} &= 1 + 1.76\rho_d + 0.37\rho_d^2 \\
3)\quad \epsilon_d^{'} &= 1 + 1.92\rho_d + 0.44\rho_d^2 \\
4)\quad \epsilon_d^{'} &= (1 + 0.845\rho_d)^2 \\
5)\quad \epsilon_d^{'} &= \begin{cases} 1 + 1.5995\rho_d + 1.861\rho_d^3, & 0 \le \rho_d \le 0.4\,\mathrm{g/cm^3} \\ \left[\left(1 - \frac{\rho_d}{0.917}\right) + 1.6134\rho_d\right]^3, & \rho_d > 0.4\,\mathrm{g/cm^3} \end{cases} \\
6)\quad \epsilon_d^{'} &= 1.018 + 1.2999\rho_d + 1.3842\rho_d^2 \\
7)\quad \epsilon_d^{'} &= \begin{cases} 1 + 1.4667 f_v + 1.435 f_v^3, & 0 \le f_v \le 0.45 \\ (1 - f_v) + 1.4759 f_v^3, & f_v > 0.45 \end{cases} \\
8)\quad \epsilon_d^{'} &= \left[(1 - f_v) \cdot (0.9974)^{\frac{1}{3}} + f_v \cdot (3.215)^{\frac{1}{3}}\right]^3
\end{aligned}
\tag{B.4}
$$

where ρ_d and f_v are the relative density (to water) and volume fraction of dry snow, respectively. In Eq. (B.4), the functions 1-6) use the relative density of dry snow as variable (shown in Fig. B.3) while 7-8) use the volume fraction of dry snow as variable.

It is seen in Fig. B.3 that overall the six interpolation functions match well, the maximum

difference occurs at $\rho_d = 0.917 \rightarrow f_v = 1$, which is mainly caused by the different choices of the permittivity of solid ice particle made by the authors.

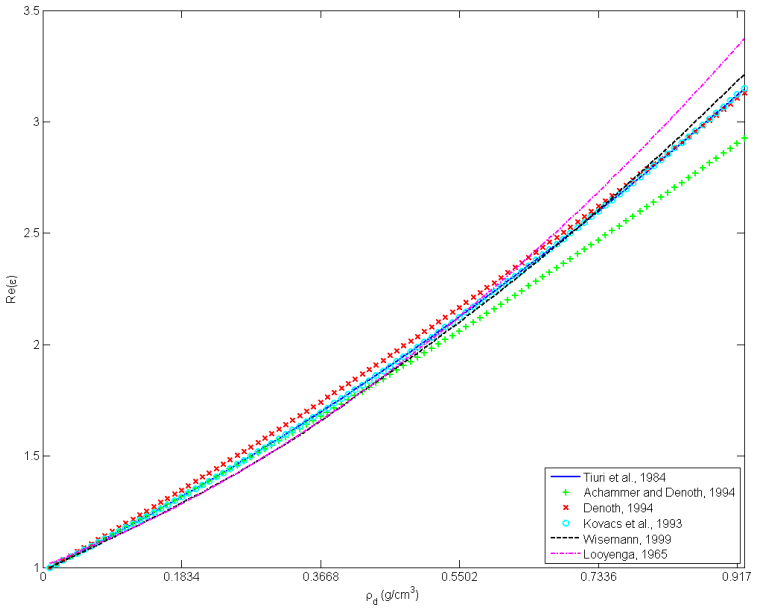


Figure B.3: The real part $\epsilon^{'}$ of the permittivity of dry snow calculated using the interpolation functions 1-6 as a function of the relative density ρ_d.

Besides the above interpolation functions, the real part of the permittivity of dry snow $\epsilon_d^{'}$ can be calculated by solving the following unified mixing formula of Sihvola and Kong, 1988 [83]:

$$\epsilon_d = \epsilon_e + \frac{f_v(\epsilon_i - \epsilon_e) + \sum_{k=1}^{3} \frac{\epsilon_a}{\epsilon_a + A_k(\epsilon_i - \epsilon_e)}}{3 - f_v(\epsilon_i - \epsilon_e) + \sum_{k=1}^{3} \frac{A_k}{\epsilon_a + A_k(\epsilon_i - \epsilon_e)}} \tag{B.5}$$

$$\epsilon_d = \epsilon_d^{'} - j\epsilon_d^{"}$$

where ϵ_e is the host medium permittivity ($\epsilon_d \rightarrow \epsilon_e$ as $f_v \rightarrow 1$), ϵ_i is the permittivity of the inclusion particles, ϵ_a is the apparent permittivity that is the one in the immediate surroundings of the inclusion particles, and A_k is the positive depolarization factor of the ellipsoids for the k^{th} principal axis, obeying

$$\sum_{k=1}^{3} A_k = 1 \tag{B.6}$$

Different mixing formula models are represented by different choices of ϵ_a varying between ϵ_e and ϵ_d:

$$\epsilon_a = \epsilon_e + a\left(\epsilon_d - \epsilon_e\right) \tag{B.7}$$

where a is a factor between 0 and 1.

For example, the generalized Maxwell-Garnett formula (also refereed as Bohren and Battan, 1982 [84]) chose $a = 0$, and the coherent potential formula set $a = 1$. The effective medium equation of Polder and Van Santen, 1946 [85] is recovered from the choice:

$$a = 1 - A_k, \quad k = 1, 2, 3 \tag{B.8}$$

Note that the value a in Eq. (B.8) is different for each term k of the sum in Eq. (B.6). According to [43], the Polder and Van Santen equation turned out to be the best description of the dry snow permittivity. In the Polder and Van Santen model, the mixing formula is further simplified by

$$A = A_1 = A_2 \tag{B.9}$$

, which is made by the assumption of spheroidal particle for which two of the three depolarization factors are equal. Note that now A is the only shape parameter, for which $A < \frac{1}{3}$ represents oblate spheroid while $\frac{1}{3} < A \leq \frac{1}{2}$ represents prolate spheroid.

According to [82],

$$A = \begin{cases} 0.1 + 0.5 f_v, & 0 < f_v < 0.33 \\ 0.18 + 3.24\left(f_v - 0.49\right)^2, & 0.33 \leq f_v < 0.71 \\ 1/3, & f_v \geq 0.71 \end{cases} \tag{B.10}$$

and

$$\epsilon_e = 1$$
$$\epsilon_i = 3.185 \tag{B.11}$$

Other choices of ϵ_e and ϵ_i made by other authors can be found in Tab. I of [82].

The real part of the permittivity of dry snow $\epsilon_d^{'}$ calculated by the interpolation functions 7) and 8), and the mixing formula by Polder and Van Santen are shown in Fig. B.4. It is seen that three models produce almost identical values and it can also be noted that all values of the real part $\epsilon_d^{'}$ in both Fig. B.4 and B.4 are close to each other.

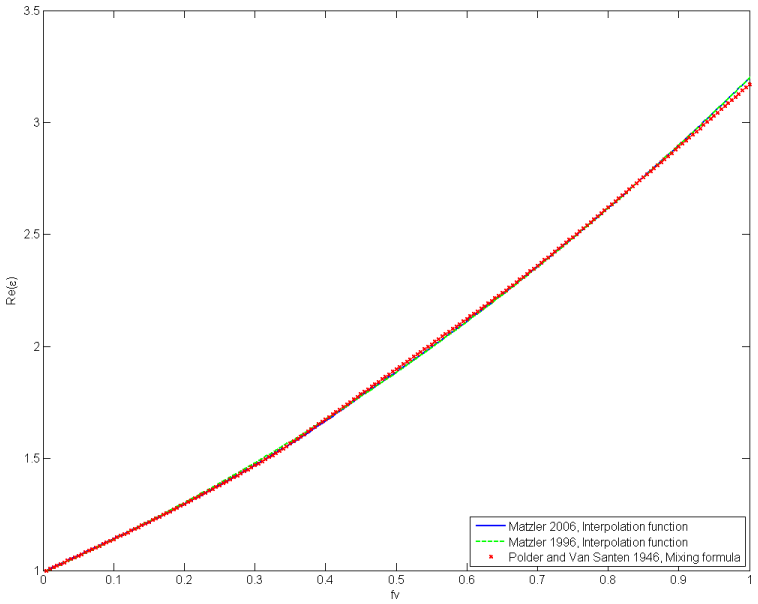


Figure B.4: The real part $\epsilon^{'}$ of dry snow calculated using the interpolation functions 7), 8), and the mixing formula (Eq. B.5) from Polder and Van Santen as a function of the volume fraction f_v.

The imaginary part of the permittivity of dry snow $\epsilon_d^{"}$ can be computed from solving the Eq. (B.5) along with using a complex value of ϵ_i: 1) Tiuri et al., 1984 [76] and 2) Matzler and Wisemann, 1999 [86] presented the following simpler expressions

$$1) \quad \epsilon_d^{"} = \left(0.48 f_v + 0.52 f_v^2\right) \cdot \epsilon_i^{"} \tag{B.12}$$

$$2) \quad \epsilon_d^{"} = \quad \sqrt{\epsilon^{'}} K^2 f_v \epsilon_i^{"} \tag{B.13}$$

to be good approximations to their data, respectively. In Eq. (B.13) K^2 is set to be 0.5 for dry snow according to [86]. Eqs. (B.12) and (B.13) give very similar results over the entire volume fraction range.

B.2.2 Permittivity of Wet Snow

Wet snow is commonly modeled as water particles mixing within a host medium of dry snow, and has the following two characteristics: 1) the water particles are not homogeneously distributed within the host medium and 2) due to the saturation of pores by water, typically the liquid water content (W) does not exceed 10% to 12%. Due to the granular nature of snow the method for permittivity measurement of wet snow using capacitance sensors, reflectometer and resonator techniques is limited to low frequencies (less than 10 GHz).

The real part of the permittivity of wet snow $\epsilon_{ws}^{'}$ can be calculated by the following interpolation functions: 1) Frolov et al., 1996 [87], 2) Achammer and Denoth, 1994 [78], 3) Denoth, 1989 [88],

$$\epsilon_{ws}^{'} = \epsilon_d^{'} + \triangle\epsilon_{ws}^{'} \Leftarrow \begin{cases} 1) \quad \triangle\epsilon_{ws}^{'} = \quad 16.7W + 42.5W^2 \\ 2) \quad \triangle\epsilon_{ws}^{'} = \quad 14.4W + 139.9W^2 \\ 3) \quad \triangle\epsilon_{ws}^{'} = \quad 20.6W + 46W^2 \end{cases} \tag{B.14}$$

and 4) the mixing formula in Eq. (B.5) with the following modified parameters by Matzler, 1987 [89]

$$\epsilon_{ws} = \epsilon_d + \sum_{k=1}^{3} \epsilon_k, \quad k = 1, 2, 3 \tag{B.15}$$

where

$$
\begin{aligned}
\epsilon_k &= \epsilon_{\infty k} + \frac{\epsilon_{sk} - \epsilon_{\infty k}}{1 - j(f/f_{0k})} \\
\epsilon_{sk} &= \frac{W}{3} \cdot \frac{\epsilon_d(\epsilon_{sw} - \epsilon_d)}{\epsilon_d + A_k(\epsilon_{sw} - \epsilon_d)} \\
\epsilon_{\infty k} &= \frac{W}{3} \cdot \frac{\epsilon_d(\epsilon_{\infty w} - \epsilon_d)}{\epsilon_d + A_k(\epsilon_{\infty w} - \epsilon_d)} \\
f_{0k} &= f_{0w}\left[1 + \frac{A_k(\epsilon_{sw} - \epsilon_{\infty w})}{\epsilon_d + A_k(\epsilon_{sw} - \epsilon_d)}\right]
\end{aligned}
$$

where ϵ_{sw}, $\epsilon_{\infty w}$, and f_{0w} are the static and infinite frequency permittivities, and relaxation frequency of pure water, respectively, and $A_1 = A_2 = 0.4975$, $A_3 = 0.005$.

Results of the above four models are shown in Fig. B.5.

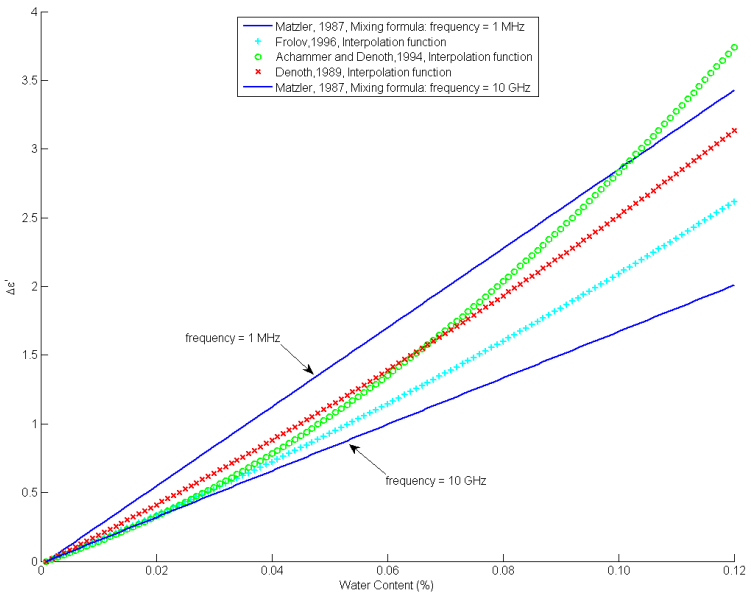


Figure B.5: The real part $\epsilon^{'}$ of wet snow calculated using the three interpolation functions and one mixing formula equation, plotted as a function of the liquid water content.

The three interpolation functions have been fitted by their own experiment data within less than one percent mean error. It is seen that the results of the three functions fall into a reasonable small range, which is bounded by the mixing formula model evaluated at 1 MHz and 10 GHz. Currently, the relation between the imaginary part of the permittivity of wet snow $\epsilon^{"}_{ws}$ and liquid water content needs more reliable explanation.

B.3 Sea Ice

The permittivity model of Ray, 1972 [90] is commonly used for sea ice in microwave radiative transfer calculations. The model is in a form of the Debye relaxation law as follows:

$$\epsilon_{si}^{'} = \epsilon_{\infty} + \frac{(\epsilon_s - \epsilon_{\infty})\left[1 + (\lambda_s/\lambda)^{1-\alpha}\sin(\alpha\pi/2)\right]}{1 + 2(\lambda_s/\lambda)^{1-\alpha}\sin(\alpha\pi/2) + (\lambda_s/\lambda)^{2(1-\alpha)}} \tag{B.16}$$

$$\epsilon_{si}^{''} = \frac{(\epsilon_s - \epsilon_{\infty})(\lambda_s/\lambda)^{1-\alpha}\cos(\alpha\pi/2)}{1 + 2(\lambda_s/\lambda)^{1-\alpha}\sin(\alpha\pi/2) + (\lambda_s/\lambda)^{2(1-\alpha)}} + \frac{\sigma\lambda}{18.8496 \times 10^{10}} \tag{B.17}$$

where the parameters are

$$\begin{aligned}
\epsilon_{\infty} &= 3.168 \\
\alpha &= 0.288 + 0.0052T + 0.00023T^2 \\
\sigma &= 1.26\exp\{-12500/[(T + 273.15)\cdot 1.9869]\} \\
\lambda_s &= 9.990288 \times 10^{-4}\exp\{13200/[(T + 273.15)\cdot 1.9869]\} \\
\epsilon_s &= 203.168 + 2.5T + 0.15T^2
\end{aligned}$$

The results of Eqs. (B.16) and (B.17) are shown in Fig. B.6 and B.7, respectively.

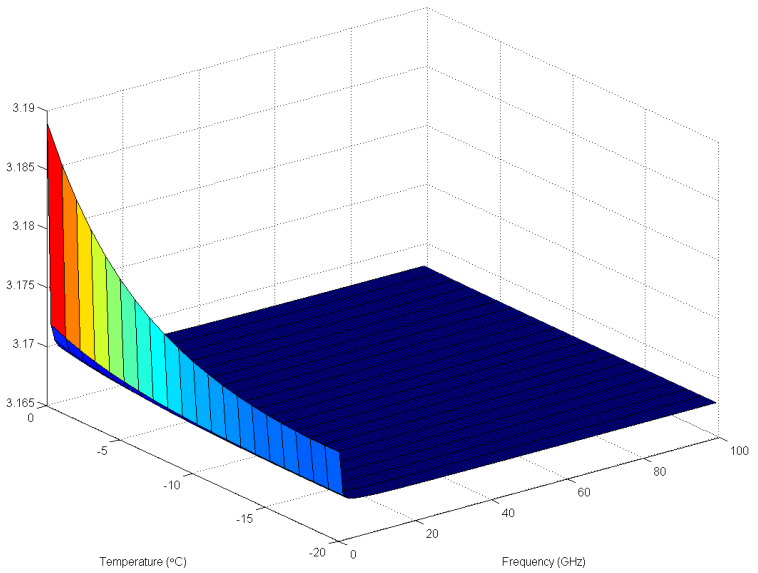


Figure B.6: The real part $\epsilon_{si}^{'}$ of permittivity of sea ice calculated as a function of frequency and temperature using the Ray, 1972 model.

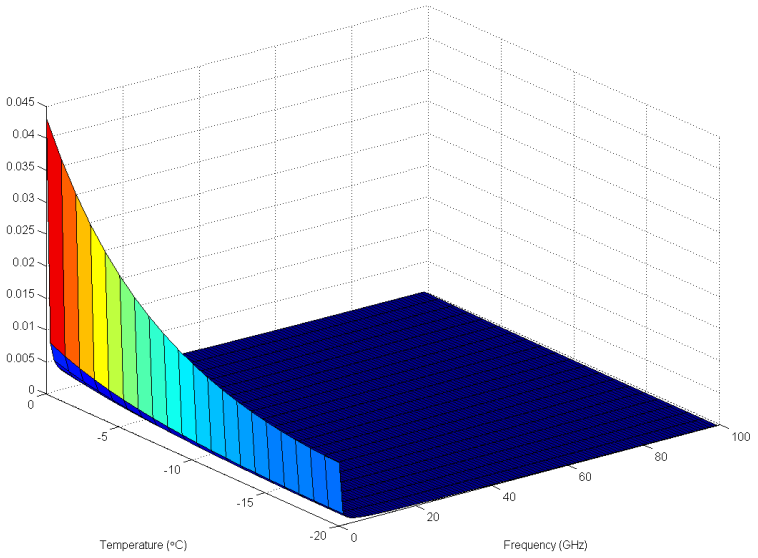


Figure B.7: The imaginary part $\epsilon_{si}^{"}$ of permittivity of sea ice calculated as functions of frequency and temperature using the Ray, 1972 model.

It is seen that both real and imaginary parts of the sea ice permittivity exhibit similar behavior: more sensitive to temperature variation than frequency variation. When frequency is above ~20

GHz, the sea ice permittivity is close to a constant.

Appendix C

Discrete Ordinate Eigenanalysis Solution

Following the notation in Chapter 2, the discretized homogeneous differential radiative transfer equation (DRTE) has following matrix form:

$$\frac{d}{dz}\begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} = \begin{bmatrix} -\overline{\overline{U}} & -\overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} \tag{C.1}$$

Let $\overline{u}'(z) = \overline{u}(z) - \overline{v}(z)$ and $\overline{v}'(z) = \overline{u}(z) + \overline{v}(z)$. The relations between $\overline{u}'(z)$, $\overline{v}'(z)$ and $\overline{u}(z)$, $\overline{v}(z)$ are

$$\begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} \tag{C.2}$$

and

$$\begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} \tag{C.3}$$

Substituting the above relations into Eq. (C.1) yields

$$
\begin{aligned}
\frac{d}{dz}\begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} &= \frac{d}{dz}\left\{ \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} \right\} \\
&= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -\overline{\overline{U}} & -\overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} \\
&= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -\overline{\overline{U}} & -\overline{\overline{D}} \\ \overline{\overline{D}} & \overline{\overline{U}} \end{bmatrix} \left\{ \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} \right\} \\
&= \begin{bmatrix} \overline{\overline{0}} & -\left(\overline{\overline{U}} + \overline{\overline{D}}\right) \\ -\left(\overline{\overline{U}} - \overline{\overline{D}}\right) & \overline{\overline{0}} \end{bmatrix} \begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix}
\end{aligned}
\tag{C.4}
$$

Let $\overline{\overline{A}} \triangleq \overline{\overline{U}} + \overline{\overline{D}}$ and $\overline{\overline{B}} \triangleq \overline{\overline{U}} - \overline{\overline{D}}$, Eq. (C.4) is written as

$$
\frac{d}{dz}\begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} = \begin{bmatrix} \overline{\overline{0}} & -\overline{\overline{A}} \\ -\overline{\overline{B}} & \overline{\overline{0}} \end{bmatrix} \begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} \tag{C.5}
$$

Note that as proven in Chapter 2, matrices $\overline{\overline{U}}$, $\overline{\overline{D}}$, $\overline{\overline{A}}$, and $\overline{\overline{B}}$ are all symmetric and positive definite.

Directly from Eq. (C.5), the first-order differentiation of $\overline{u}'(z)$ and $\overline{v}'(z)$ are

$$
\frac{d\overline{u}'(z)}{dz} = -\overline{\overline{A}}\overline{v}'(z) \tag{C.6}
$$

$$
\frac{d\overline{v}'(z)}{dz} = -\overline{\overline{B}}\overline{u}'(z) \tag{C.7}
$$

Using Eq. (C.5) again the second-order differentiation of $\overline{u}'(z)$ and $\overline{v}'(z)$ are

$$
\frac{d^2\overline{u}'(z)}{dz^2} = -\overline{\overline{A}}\frac{d\overline{v}'(z)}{dz} = -\overline{\overline{A}}\left[-\overline{\overline{B}}\overline{u}'(z)\right] = \overline{\overline{A}}\,\overline{\overline{B}}\overline{u}'(z) \tag{C.8}
$$

$$
\frac{d^2\overline{v}'(z)}{dz^2} = -\overline{\overline{B}}\frac{d\overline{u}'(z)}{dz} = -\overline{\overline{B}}\left[-\overline{\overline{A}}\overline{v}'(z)\right] = \overline{\overline{B}}\,\overline{\overline{A}}\overline{v}'(z) \tag{C.9}
$$

The general solution to the above second-order differentiation equations for $\overline{u}'(z)$ and $\overline{v}'(z)$ is

$$\overline{u}'(z) = e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_1 + e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_2 \tag{C.10}$$

$$\overline{v}'(z) = e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 + e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \tag{C.11}$$

where matrices $\overline{\overline{C}}_{1,2,3,4}$ are unknown coefficient matrices.

Substituting Eq. (C.10) into the left side of Eq. (C.6) and Eq. (C.11) into the right side of Eq. (C.6) yields

$$\begin{aligned} \frac{d\overline{u}'(z)}{dz} &= \sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_1 - \sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_2 \\ &= -\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 - \overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \end{aligned} \tag{C.12}$$

From Eq. (C.12), let

$$\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_1 = -\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3$$

$$\Rightarrow \overline{\overline{C}}_1 = -e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 \tag{C.13}$$

and let

$$\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{C}}_2 = \overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4$$

$$\Rightarrow \overline{\overline{C}}_2 = e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \tag{C.14}$$

Substituting Eqs. (C.13) and (C.14) into Eqs. (C.10) and (C.11) gives

$$\begin{aligned} \overline{u}'(z) = &-e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 \\ &+e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \\ = &-\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 + \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \\ \overline{v}'(z) = &\ e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 + e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \end{aligned} \tag{C.15}$$

The above expressions of $\overline{u}'(z)$ and $\overline{v}'(z)$ can be arranged in the following matrix equation:

$$\begin{aligned}\begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} &= \begin{bmatrix} \overline{\overline{c}} & -\overline{\overline{s}}\,\overline{\overline{A}} \\ -\overline{\overline{B}}\,\overline{\overline{s}} & \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} \overline{u}'(0) \\ \overline{v}'(0) \end{bmatrix} \\ &= \begin{bmatrix} \overline{\overline{c}}\,\overline{u}'(0) - \overline{\overline{s}}\,\overline{\overline{A}}\,\overline{v}'(0) \\ -\overline{\overline{B}}\,\overline{\overline{s}}\,\overline{u}'(0) + \overline{\overline{c}}^{T}\overline{v}'(0) \end{bmatrix}\end{aligned} \tag{C.16}$$

where from Eq. (C.15),

$$\begin{aligned}\overline{u}'(0) &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\left(\overline{\overline{C}}_4 - \overline{\overline{C}}_3\right) \\ \overline{v}'(0) &= \overline{\overline{C}}_3 + \overline{\overline{C}}_4\end{aligned}$$

and

$$\overline{\overline{c}} \triangleq \cosh\left(\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z\right) = \frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z} + e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}$$

$$\overline{\overline{s}} \triangleq \sinh\left(\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z\right)\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} = \frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z} - e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}$$

Proof of Eq. (C.16) being identical to Eq. (C.15) is given in the following two parts:

1) $\overline{u}'(z)$ in Eq. (C.16) can be explicitly expanded as

$$\begin{aligned}\overline{\overline{c}}\,\overline{u}'(0) - \overline{\overline{s}}\,\overline{\overline{A}}\,\overline{v}'(0) = &\ \tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}+e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\left(\overline{\overline{C}}_4 - \overline{\overline{C}}_3\right) \\ &-\tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}-e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\left(\overline{\overline{C}}_3 + \overline{\overline{C}}_4\right) \\ = &\ \tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4 - \tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 \\ &+\tfrac{e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4 - \tfrac{e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 \\ &-\tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4 - \tfrac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 \\ &+\tfrac{e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4 + \tfrac{e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\cdot\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 \\ = &\ -e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 + e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4\end{aligned} \tag{C.17}$$

Let $\overline{\overline{A}}_1 \triangleq \overline{\overline{A}}\,\overline{\overline{B}}$ and $\overline{\overline{B}}_1 \triangleq \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}$, thus

$$e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} = e^{\overline{\overline{B}}_1\overline{\overline{A}}_1 z}\overline{\overline{B}}_1 \tag{C.18}$$

Since the above matrices are all symmetric, applying the following matrix identity

$$g\left(\overline{\overline{B}}\,\overline{\overline{A}}\right)\overline{\overline{B}} = \overline{\overline{B}}g\left(\overline{\overline{A}}\,\overline{\overline{B}}\right) \tag{C.19}$$

where g is an arbitrary analytical function, to Eq. (C.18) yields:

$$\begin{aligned} e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} &= e^{\overline{\overline{B}}_1\overline{\overline{A}}_1 z}\overline{\overline{B}}_1 \\ &= \overline{\overline{B}}_1 e^{\overline{\overline{A}}_1\overline{\overline{B}}_1 z} \\ &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z} \end{aligned} \tag{C.20}$$

Hence, in Eq. (C.17)

$$\begin{aligned} \left[e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\right]\overline{\overline{A}} &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{A}} \\ &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\left[e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{A}}\right] \\ &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z} \end{aligned} \tag{C.21}$$

and similarly,

$$\begin{aligned} e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}} &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}} e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\overline{\overline{A}} \\ &= \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z} \end{aligned} \tag{C.22}$$

Substituting Eqs. (C.21) and (C.22) into Eq. (C.17) gives

$$\begin{aligned} \overline{\overline{c}}\,\overline{u}'(0) - \overline{\overline{s}}\,\overline{\overline{A}}\,\overline{v}'(0) &= -e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_3 + e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\,\overline{\overline{C}}_4 \\ &= -\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3 + \left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4 \end{aligned} \tag{C.23}$$

2) $\overline{v}'(z)$ in Eq. (C.16) can be explicitly expanded as

$$\begin{aligned} -\overline{\overline{B}}\,\overline{\overline{s}}\,\overline{u}'(0) + \overline{\overline{c}}^T\overline{v}'(0) = &-\overline{\overline{B}}\frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z} - e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}\left(\overline{\overline{C}}_4 - \overline{\overline{C}}_3\right) \\ &+\left(\frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z} + e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\right)^T\left(\overline{\overline{C}}_3 + \overline{\overline{C}}_4\right) \end{aligned} \tag{C.24}$$

Using the matrix identity in Eq. (C.19), the following matrix identity is hold:

$$\overline{\overline{B}}g\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)=g\left(\overline{\overline{B}}\,\overline{\overline{A}}\right)\overline{\overline{B}}$$

$$\Rightarrow\overline{\overline{B}}g\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)\overline{\overline{B}}^{-1}=g\left(\overline{\overline{B}}\,\overline{\overline{A}}\right)\overline{\overline{B}}\,\overline{\overline{B}}^{-1}=g\left(\overline{\overline{B}}\,\overline{\overline{A}}\right)\tag{C.25}$$

Using the above matrix identity yields

$$\begin{aligned}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{-\frac{1}{2}}\overline{\overline{A}}= &\ \overline{\overline{B}}^{-\frac{1}{2}}\overline{\overline{A}}^{-\frac{1}{2}}\overline{\overline{B}}^{-\frac{1}{2}}\overline{\overline{A}}^{-\frac{1}{2}}\overline{\overline{A}}\\ = &\ \overline{\overline{B}}^{-\frac{1}{2}}\left(\overline{\overline{A}}^{-\frac{1}{2}}\overline{\overline{B}}^{-\frac{1}{2}}\overline{\overline{A}}^{\frac{1}{2}}\right)\\ = &\ \overline{\overline{B}}^{-\frac{1}{2}}\overline{\overline{B}}^{-\frac{1}{2}}=\overline{\overline{B}}^{-1}\end{aligned}\tag{C.26}$$

Moreover, since $\overline{\overline{A}}$ and $\overline{\overline{B}}$ are both symmetric, thus

$$\left(\overline{\overline{A}}\,\overline{\overline{B}}\right)^{T}=\overline{\overline{B}}^{T}\overline{\overline{A}}^{T}=\overline{\overline{B}}\,\overline{\overline{A}}$$

$$\Rightarrow\left(\frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}+e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\right)^{T}=\frac{e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}+e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}}{2}\tag{C.27}$$

Substituting Eqs. (C.26) and (C.27) into the right side of Eq. (C.24) yields

$$\begin{aligned}-\overline{\overline{B}}\,\overline{\overline{s}}\,\overline{u}'(0)+\overline{\overline{c}}^{T}\overline{v}'(0)= &\ -\overline{\overline{B}}\frac{e^{\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}-e^{-\sqrt{\overline{\overline{A}}\,\overline{\overline{B}}}z}}{2}\overline{\overline{B}}^{-1}\left(\overline{\overline{C}}_4-\overline{\overline{C}}_3\right)+\frac{e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}+e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}}{2}\left(\overline{\overline{C}}_3+\overline{\overline{C}}_4\right)\\ = &\ -\frac{e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}-e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}}{2}\left(\overline{\overline{C}}_4-\overline{\overline{C}}_3\right)+\frac{e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}+e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}}{2}\left(\overline{\overline{C}}_3+\overline{\overline{C}}_4\right)\\ = &\ e^{\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_3+e^{-\sqrt{\overline{\overline{B}}\,\overline{\overline{A}}}z}\overline{\overline{C}}_4\end{aligned}\tag{C.28}$$

Comparing Eqs. (C.23) and (C.28) with Eq. (C.15), it is shown that Eq. (C.16) produces identical results to that of Eq. (C.15).

Finally, Eq. (C.15) can be rearranged by reverting the solution basis from $\overline{u}'(z)$ and $\overline{v}'(z)$ to $\overline{u}(z)$ and $\overline{v}(z)$ via following steps:

$$\begin{bmatrix} \overline{u}'(z) \\ \overline{v}'(z) \end{bmatrix} = \begin{bmatrix} \overline{\overline{c}} & -\overline{\overline{s}}\,\overline{\overline{A}} \\ -\overline{\overline{B}}\,\overline{\overline{s}} & \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} \overline{u}'(0) \\ \overline{v}'(0) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} = \begin{bmatrix} \overline{\overline{c}} & -\overline{\overline{s}}\,\overline{\overline{A}} \\ -\overline{\overline{B}}\,\overline{\overline{s}} & \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}(0) \\ \overline{v}(0) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \overline{u}(z) \\ \overline{v}(z) \end{bmatrix} = \tfrac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \overline{\overline{c}} & -\overline{\overline{s}}\,\overline{\overline{A}} \\ -\overline{\overline{B}}\,\overline{\overline{s}} & \overline{\overline{c}}^{T} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \overline{u}(0) \\ \overline{v}(0) \end{bmatrix} \tag{C.29}$$

Eq. (C.29) is the solution for the homogeneous DRTE used in Chapter 2 and DOTLRT [22].

Appendix D

Refractivity Adjusted Reflection and Transmission Matrices

Alternatively, the planar interface between medium layers n and $n+1$ can be described as a Fresnel-Snell transition layer (indexed by $n+0.5$), illustrated in Fig. D.1. The Fresnel-Snell transition layer has following two characteristics: 1) infinitely thin and 2) zero emission. The transition layer only redistribute the incident radiation streams according to Fresnel's and Snell's law. The Fresnel reflectivity and transmissivity matrices (respectively) are denoted by lower-case $\bar{\bar{r}}^{(n+0.5)\uparrow\downarrow}$ and $\bar{\bar{t}}^{(n+0.5)\uparrow\downarrow}$, and annotated with arrows $\uparrow$ and $\downarrow$ to indicate the corresponding directions of the incident streams.

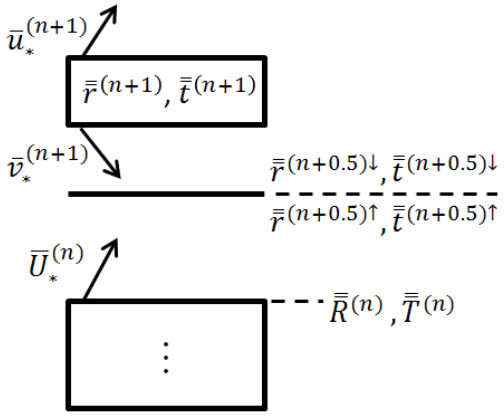


Figure D.1: A planar multilayer stack with the Fresnel-Snell transition layer.

Similar as the procedure used within DOTLRT, Eqs. (59-67) in [22], the refractivity adjusted reflection and transmission matrices are derived from the following two cases.

Case 1: the refractivity adjusted reflection and transmission matrices, and upwelling radiation

stream vector evaluated at the top of the transition layer: $\overline{\overline{R}}^{(n+0.5)}$, $\overline{\overline{T}}^{(n+0.5)}$ and $\overline{U}_*^{(n+0.5)}$.

1a) the upwelling radiation streams $\overline{U}_*^{(n+0.5)}$ (Fig. D.2)

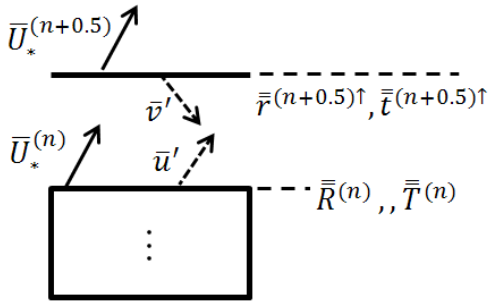


Figure D.2: Illustration of the upwelling radiation streams $\overline{U}_*^{(n+0.5)}$.

At the boundary between the n-layer stack and the transition layer $(n+0.5)$, there will be present the upwelling thermal radiation stream vector $\overline{U}_*^{(n)}$ along with additional stream vectors $\overline{u}'$ and $\overline{v}'$ due to the multiple reflections between the stack and layer. These stream vectors must satisfy the following equations:

$$\overline{u}' = \overline{\overline{R}}^{(n)}\overline{v}' \tag{D.1}$$

$$\overline{v}' = \bar{\bar{r}}^{(n+0.5)\uparrow}\left(\overline{U}_*^{(n)} + \overline{u}'\right) \tag{D.2}$$

$$\overline{U}_*^{(n+0.5)} = \bar{\bar{t}}^{(n+0.5)\uparrow}\left(\overline{U}_*^{(n)} + \overline{u}'\right) \tag{D.3}$$

Substituting Eq. (D.2) into Eq. (D.1D.2) yields

$$\begin{aligned} \overline{u}' &= \overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\left(\overline{U}_*^{(n)} + \overline{u}'\right) \\ \Rightarrow \overline{u}' &= \left(\bar{\bar{I}} - \overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\overline{U}_*^{(n)} \end{aligned} \tag{D.4}$$

Substituting Eqs. (D.4) and (D.2) into Eq. (D.1) yields

$$\left(\bar{\bar{I}} - \overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\overline{U}_*^{(n)} = \overline{\overline{R}}^{(n)}\bar{\bar{r}}^{(n+0.5)\uparrow}\left(\overline{U}_*^{(n)} + \overline{u}'\right) \tag{D.5}$$

Thus,

$$\overline{U}_*^{(n)} + \overline{u}' = \left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)\overline{U}_*^{(n)} \tag{D.6}$$

Since,

$$\begin{aligned} & \left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right) \\ = & \left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\sum_{n=0}^{\infty}\left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{n}\left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right) \\ = & \sum_{n=0}^{\infty}\left(\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{n} = \left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1} \end{aligned} \tag{D.7}$$

where $\left\|\overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right\| < 1$.

Using the above result Eq. (D.6) is simplified as

$$\overline{U}_*^{(n)} + \overline{u}' = \left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{U}_*^{(n)} \tag{D.8}$$

Substituting Eq. (D.8) into (D.3) gives

$$\overline{U}_*^{(n+0.5)} = \overline{\overline{t}}^{(n+0.5)\uparrow}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{U}_*^{(n)} \tag{D.9}$$

1b) the refractivity adjusted reflection matrix (downward) at the top the transition layer $\overline{\overline{R}}^{(n+0.5)}$

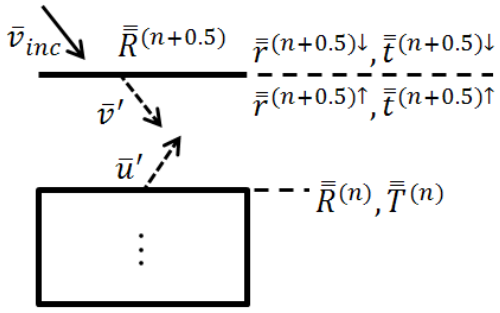


Figure D.3: Illustration of the refractivity adjusted reflection matrix (downward) $\overline{\overline{R}}^{(n+0.5)}$.

Assuming a downwelling incident radiation stream vector $\overline{v}_{inc}$ (from above the transition layer, Fig. D.3), the following equations must hold:

$$\begin{aligned} \overline{u}' &= \overline{\overline{R}}^{(n)}\overline{v}' \\ \overline{v}' &= \overline{\overline{t}}^{(n+0.5)\downarrow}\overline{v}_{inc} + \overline{\overline{r}}^{(n+0.5)\uparrow}\overline{u}' \\ \overline{\overline{R}}^{(n+0.5)}\overline{v}_{inc} &= \overline{\overline{r}}^{(n+0.5)\downarrow}\overline{v}_{inc} + \overline{\overline{t}}^{(n+0.5)\uparrow}\overline{u}' \end{aligned} \tag{D.10}$$

Similarly, solving the three equations in (D.10) together yields

$$\overline{\overline{R}}^{(n+0.5)} = \overline{\overline{r}}^{(n+0.5)\downarrow} + \overline{\overline{t}}^{(n+0.5)\uparrow}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{\overline{R}}^{(n)}\overline{\overline{t}}^{(n+0.5)\downarrow} \tag{D.11}$$

1c) the refractivity adjusted transmission matrix (upward) at the top the transition layer $\overline{\overline{T}}^{(n+0.5)}$

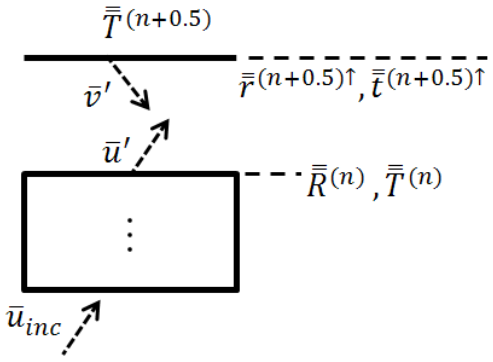


Figure D.4: Illustration of the refractivity adjusted transmission matrix (upward) $\overline{\overline{T}}^{(n+0.5)}$.

Assuming a upwelling incident radiation stream vector $\overline{u}_{inc}$ (from below the n-layer stack, Fig. D.4), the following equations must be satisfied:

$$\begin{aligned} \overline{u}' &= \overline{\overline{R}}^{(n)}\overline{v}' + \overline{\overline{T}}^{(n)}\overline{u}_{inc} \\ \overline{v}' &= \overline{\overline{r}}^{(n+0.5)\uparrow}\overline{u}' \\ \overline{\overline{T}}^{(n+0.5)}\overline{u}_{inc} &= \overline{\overline{t}}^{(n+0.5)\uparrow}\overline{u}' \end{aligned} \tag{D.12}$$

Solving the three equations in (D.12) together gives

$$\overline{\overline{T}}^{(n+0.5)} = \overline{\overline{t}}^{(n+0.5)\uparrow}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n)}\overline{\overline{r}}^{(n+0.5)\uparrow}\right)^{-1}\overline{\overline{T}}^{(n)} \tag{D.13}$$

Case 2: the refractivity adjusted reflection and transmission matrices, and upwelling radiation stream vector evaluated at the top of the $(n+1)$-layer stack: $\overline{\overline{R}}^{(n+1)}$, $\overline{\overline{T}}^{(n+1)}$ and $\overline{U}_*^{(n+1)}$.

The solutions to case 2 are similar to that of the previous case.

2a) the upwelling radiation stream vector $\overline{U}_*^{(n+1)}$

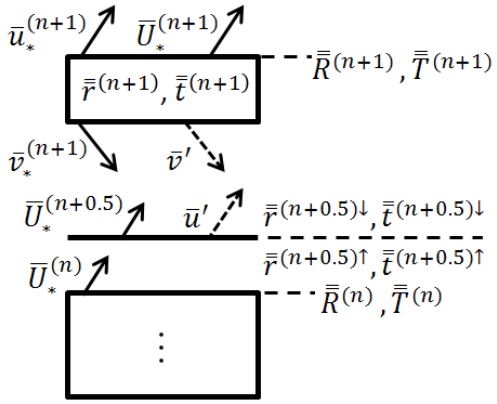


Figure D.5: Illustration of the refractivity adjusted reflection and transmission matrices, $\overline{\overline{R}}^{(n+1)}$ and $\overline{\overline{T}}^{(n+1)}$, and upwelling radiation stream $\overline{U}_*^{(n+1)}$.

The stream vectors in Fig. D.5 must satisfy the following equations:

$$\begin{aligned}
\overline{u}' &= \overline{\overline{R}}^{(n+0.5)}\left(\overline{v}' + \overline{v}_*^{(n+1)}\right) \\
\overline{v}' &= \overline{\overline{r}}^{(n+1)}\left(\overline{u}' + \overline{U}_*^{(n+0.5)}\right) \\
\overline{U}_*^{(n+1)} &= \overline{u}_*^{(n+1)} + \overline{\overline{t}}^{(n+1)}\left(\overline{u}' + \overline{U}_*^{(n+0.5)}\right)
\end{aligned} \tag{D.14}$$

Solving the three equations in (E.6) together gives

$$\overline{U}_*^{(n+1)} = \overline{u}_*^{(n+1)} + \overline{\overline{t}}^{(n+1)}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n+0.5)}\overline{\overline{r}}^{(n+1)}\right)^{-1}\left(\overline{U}_*^{(n+0.5)} + \overline{\overline{R}}^{(n+0.5)}\overline{v}_*^{(n+1)} +\right) \tag{D.15}$$

2b-2c) Similarly, from following two equation sets

$$\begin{aligned} \overline{u}' &= \overline{\overline{R}}^{(n+0.5)}\overline{v}' \\ \overline{v}' &= \overline{\overline{t}}^{(n+1)}\overline{v}_{inc} + \overline{\overline{r}}^{(n+1)}\overline{u}' \\ \overline{\overline{R}}^{(n+1)}\overline{v}_{inc} &= \overline{\overline{r}}^{(n+1)}\overline{v}_{inc} + \overline{\overline{t}}^{(n+1)}\overline{u}' \end{aligned} \tag{D.16}$$

and

$$\begin{aligned} \overline{u}' &= \overline{\overline{R}}^{(n+0.5)}\overline{v}' + \overline{\overline{T}}^{(n+0.5)}\overline{u}_{inc} \\ \overline{v}' &= \overline{\overline{r}}^{(n+1)}\overline{u}' \\ \overline{\overline{T}}^{(n+1)}\overline{u}_{inc} &= \overline{\overline{t}}^{(n+1)}\overline{u}' \end{aligned} \tag{D.17}$$

One can solve $\overline{\overline{R}}^{(n+1)}$ and $\overline{\overline{T}}^{(n+1)}$, respectively.

$$\begin{aligned} \overline{\overline{R}}^{(n+1)} &= \overline{\overline{r}}^{(n+1)} + \overline{\overline{t}}^{(n+1)}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n+0.5)}\overline{\overline{r}}^{(n+1)}\right)^{-1}\overline{\overline{R}}^{(n+0.5)}\overline{\overline{t}}^{(n+1)} \\ \overline{\overline{T}}^{(n+1)} &= \overline{\overline{t}}^{(n+1)}\left(\overline{\overline{I}} - \overline{\overline{R}}^{(n+0.5)}\overline{\overline{r}}^{(n+1)}\right)^{-1}\overline{\overline{T}}^{(n+0.5)} \end{aligned} \tag{D.18}$$

The above two cases are derived based on a upward recursion. The cases of the downward recursion can be derived in an analogous manner.

Applying the above results to an isolated 2-layer stack, which does not include any refracting interface, we have:

$$\begin{aligned} \overline{\overline{R}}^{(0.5)} &= \overline{\overline{0}} \\ \overline{\overline{T}}^{(0.5)} &= \overline{\overline{I}} \\ \overline{U}_*^{(0.5)} &= \overline{U}_*^{(0)} \end{aligned} \tag{D.19}$$

Thus,

$$\begin{aligned} \overline{\overline{R}}^{(1)} &= \overline{\overline{r}}^{(1)} \\ \overline{\overline{T}}^{(1)} &= \overline{\overline{t}}^{(1)} \\ \overline{U}_*^{(1)} &= \overline{u}_*^{(1)} + \overline{\overline{t}}^{(1)}\overline{U}_*^{(0)} \end{aligned} \tag{D.20}$$

Since $\overline{\overline{r}}^{(1.5)\uparrow\downarrow} = \overline{\overline{0}}$ and $\overline{\overline{t}}^{(1.5)\uparrow\downarrow} = \overline{\overline{I}}$,

$$\begin{aligned}
\overline{\overline{R}}^{(1.5)} &= \overline{\overline{R}}^{(1)} \\
\overline{\overline{T}}^{(1.5)} &= \overline{\overline{T}}^{(1)} \\
\overline{U}_*^{(1.5)} &= \overline{U}_*^{(1)}
\end{aligned} \tag{D.21}$$

and then

$$\begin{aligned}
\overline{\overline{R}}^{(2)} &= \overline{\overline{r}}^{(2)} + \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1.5)} \overline{\overline{r}}^{(2)} \right)^{-1} \overline{\overline{R}}^{(1.5)} \overline{\overline{t}}^{(2)} = \overline{\overline{r}}^{(2)} + \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1)} \overline{\overline{r}}^{(2)} \right)^{-1} \overline{\overline{R}}^{(1)} \overline{\overline{t}}^{(2)} \\
\overline{\overline{T}}^{(2)} &= \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1.5)} \overline{\overline{r}}^{(n+1)} \right)^{-1} \overline{\overline{T}}^{(1.5)} = \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1)} \overline{\overline{r}}^{(n+1)} \right)^{-1} \overline{\overline{T}}^{(1)} \\
\overline{U}_*^{(2)} &= \overline{u}_*^{(2)} + \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1.5)} \overline{\overline{r}}^{(2)} \right)^{-1} \left(\overline{U}_*^{(1.5)} + \overline{\overline{R}}^{(1.5)} \overline{v}_*^{(2)} \right) \\
&= \overline{u}_*^{(2)} + \overline{\overline{t}}^{(2)} \left(\overline{\overline{I}} - \overline{\overline{R}}^{(1)} \overline{\overline{r}}^{(2)} \right)^{-1} \left(\overline{U}_*^{(1)} + \overline{\overline{R}}^{(1)} \overline{v}_*^{(2)} \right)
\end{aligned} \tag{D.22}$$

The results in Eq. (D.22) are identical to that of the DOTLRT and the method described in Chapter 3 of this thesis.

Appendix E

First Order Perturbation Theory

For an $n \times n$ symmetric phase matrix $\overline{\overline{A}}$, it can be expressed as

$$\overline{\overline{A}} = \overline{\overline{M}}_1 \overline{\overline{\Lambda}}_1 \overline{\overline{M}}_1^T \tag{E.1}$$

where $\overline{\overline{M}}_1$ is a full matrix consisting of eigenvectors $\overline{X}_i$, $i = 1, \cdots n$ and $\overline{\overline{\Lambda}}_1$ is a diagonal matrix and contains eigenvalues λ_i, $i = 1, \cdots n$.

Using the above definitions,

$$\overline{\overline{A}}\,\overline{X}_k = \lambda_k \overline{X}_k,\ k \in [1, n] \tag{E.2}$$

The first order perturbation yields

$$\overline{\overline{A}} = \overline{\overline{A}}_0 + \varepsilon \dot{\overline{\overline{A}}} \tag{E.3}$$

$$\overline{X}_k = \overline{X}_{k0} + \varepsilon \dot{\overline{X}}_k \tag{E.4}$$

$$\lambda_k = \lambda_{k0} + \varepsilon \dot{\lambda}_k \tag{E.5}$$

where ε is a variable parameter.

For the unperturbed matrix $\overline{\overline{A}}_0$, Eq. (E.2) yields

$$\overline{\overline{A}}_0 \overline{u}_k = \lambda_k^0 \overline{u}_k,\ k \in [1, n] \tag{E.6}$$

where $\overline{u}_k$ and λ_k^0 are the known eigenvectors and eigenvalues, respectively.

In the limit $\varepsilon \to 0$, the perturbation vanishes so that $\overline{X}_k$ and λ_k reduce to the unperturbed solutions:

$$\begin{aligned} \overline{X}_k &\to \quad \overline{X}_{k0} \equiv \overline{u}_k \\ \lambda_k &\to \quad \lambda_{k0} \equiv \lambda_k^0 \end{aligned} \tag{E.7}$$

Substituting Eqs. (E.3-E.5) into Eq. (E.2) yields

$$\left(\overline{\overline{A}}_0 + \varepsilon \dot{\overline{\overline{A}}}\right)\left(\overline{X}_{k0} + \varepsilon \dot{\overline{X}}_k\right) = \left(\lambda_{k0} + \varepsilon \dot{\lambda}_k\right)\left(\overline{X}_{k0} + \varepsilon \dot{\overline{X}}_k\right) \tag{E.8}$$

$$\overline{\overline{A}}_0 \overline{X}_{k0} + \varepsilon \overline{\overline{A}}_0 \dot{\overline{X}}_k + \varepsilon \dot{\overline{\overline{A}}} \overline{X}_{k0} + \varepsilon^2 \dot{\overline{\overline{A}}} \dot{\overline{X}}_k = \lambda_{k0} \overline{X}_{k0} + \varepsilon \lambda_{k0} \dot{\overline{X}}_k + \varepsilon \dot{\lambda}_k \overline{X}_{k0} + \varepsilon^2 \dot{\lambda}_k \dot{\overline{X}}_k \tag{E.9}$$

In Eq. (E.9), neglecting the 2^{nd} order terms and balancing the remaining terms yields:

$$\overline{\overline{A}}_0 \overline{X}_{k0} = \lambda_{k0} \overline{X}_{k0} \tag{E.10}$$

$$\overline{\overline{A}}_0 \dot{\overline{X}}_k + \dot{\overline{\overline{A}}} \overline{X}_{k0} = \lambda_{k0} \dot{\overline{X}}_k + \dot{\lambda}_k \overline{X}_{k0} \tag{E.11}$$

where Eq. (E.10) represents a restatement of the unperturbed starting point and again

$$\overline{X}_{k0} = \overline{u}_k,\ \ \lambda_{k0} = \lambda_k^0 \tag{E.12}$$

The $\dot{\overline{X}}_k$ can be expanded in the unperturbed eigenvectors $\overline{u}_k$:

$$\dot{\overline{X}}_k = \sum_j c_{jk}^{(1)} \overline{u}_j \tag{E.13}$$

Substituting Eq. (E.13) into Eq. (E.11) and remembering that $\overline{X}_{k0} = \overline{u}_k$ yields

$$\overline{\overline{A}}_0 \sum_j c_{jk}^{(1)} \overline{u}_j + \dot{\overline{\overline{A}}} \overline{u}_k = \lambda_k^0 \sum_j c_{jk}^{(1)} \overline{u}_j + \dot{\lambda}_k \overline{u}_k \tag{E.14}$$

Since the $\overline{u}_j$ are eigenvectors of $\overline{\overline{A}}_0$ with corresponding eigenvalue λ_k^0, Eq. (E.14) is expressed as

$$\sum_j \lambda_j^0 c_{jk}^{(1)} \overline{u}_j + \dot{\overline{\overline{A}}} \overline{u}_k = \sum_j \lambda_k^0 c_{jk}^{(1)} \overline{u}_j + \dot{\lambda}_k \overline{u}_k \tag{E.15}$$

and multiplying from the left by $\overline{u}_k^T$ in (E.16) gives

$$\dot{\overline{\overline{A}}} \overline{u}_k = \dot{\lambda}_k \overline{u}_k \tag{E.16}$$

$$\overline{u}_k^T \dot{\overline{\overline{A}}} \overline{u}_k = \overline{u}_k^T \dot{\lambda}_k \overline{u}_k \tag{E.17}$$

In Eq. (E.17), since $\dot{\lambda}_k$ is only a number and $\overline{u}_k$ are orthogonal sets, $\overline{u}_k^T \dot{\lambda}_k \overline{u}_k = \dot{\lambda}_k \overline{u}_k^T \overline{u}_k = \dot{\lambda}_k$. Hence, Eq. (E.17) yields

$$\dot{\lambda}_k = \left(\overline{u}_k^T \dot{\lambda}_k \overline{u}_k \right)_{kk} \tag{E.18}$$

Similarly, multiplying from the left by $\overline{u}_l^T$ in Eq. (E.14) gives

$$\overline{u}_l^T \lambda_j^0 \sum_j c_{jk}^{(1)} \overline{u}_j + \overline{u}_l^T \dot{\overline{\overline{A}}} \overline{u}_k = \overline{u}_l^T \lambda_k^0 \sum_j c_{jk}^{(1)} \overline{u}_j + \overline{u}_l^T \dot{\lambda}_k \overline{u}_k \tag{E.19}$$

In Eq. (E.19),

$$\overline{u}_l^T \lambda_j^0 \sum_j c_{jk}^{(1)} \overline{u}_j = \lambda_l^0 c_{lk}^{(1)} \tag{E.20}$$

$$\overline{u}_l^T \lambda_k^0 \sum_j c_{jk}^{(1)} \overline{u}_j = \lambda_k^0 c_{lk}^{(1)} \tag{E.21}$$

$$\overline{u}_l^T \dot{\lambda}_k \overline{u}_k = 0, \;\; l \neq k \tag{E.22}$$

Substituting Eqs. (E.20-E.22) into Eq. (E.19) yields

$$\lambda_l^0 c_{lk}^{(1)} + \overline{u}_l^T \dot{\overline{\overline{A}}} \overline{u}_k = \lambda_k^0 c_{lk}^{(1)} \tag{E.23}$$

The $c_{lk}^{(1)}$ can be solved from Eq. (E.23),

$$c_{lk}^{(1)} = \frac{\overline{u}_l^T \dot{\overline{\overline{A}}} \overline{u}_k}{\lambda_k^0 - \lambda_l^0}, \;\; l \neq k \tag{E.24}$$

Substituting Eq. (E.24) into Eq. (E.13) yields

$$\dot{\overline{X}}_k = \sum_{l \neq k} \frac{\overline{u}_l^T \dot{\overline{\overline{A}}} \overline{u}_k}{\lambda_k^0 - \lambda_l^0} \overline{u}_l \tag{E.25}$$

Appendix F

UMRT Jacobian Program Listing

The source codes for the entire UMRT Jacobian formulation are presented in this appendix. All UMRT Jacobian codes (51 major programs in total) are written in MATLAB. The two programs for computing the effective propagation constant and the Percus-Yevick pair distribution function in the DMRT-QCA theory were originally written by Drs. C. Chen and K.H. Ding, respectively in 1998. Another two programs for computing the Mie coefficients were originally written by Dr. C. Matzler in 2002. The above four programs were modified by the author to fulfil the purpose of this research. Moreover, there are two programs written by Dr. S. Kumar in 2008 for computing the permittivity of liquid water and water-ice. All other programs were written entirely by the author.

```
%%% Program Name: UMRT_Jac_Main_Ref_v2
%%% Description:
%%% 1. This code is the main program for the entire UMRT Jacobian calculation
%%%    for a Multilayer stack with refractive boundaries
%%% 2. INPUT: a. profile: n by m matrix, where n is the number of layers used
%%%                       for the multilayer model and m is the number of the
%%%                       parameters needed for each individual layer
%%%           b. Gauss-Legendre nodes and weights: xk and wx, which are
%%%                       computed by weights_yak.m and glpt.m
%%%    OUTPUT: all properties including:
%%%           a. up- and down- welling brightness temperature vectors and
%%%              reflection and transmission matrices of each every single
%%%              layers and multilayer stack,
%%%           b. derivatives of all above quantities w.r.t. (currently) five
%%%              parameters: ks, ka, To, tT, d.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

profile = [10 3 1 0.1 270 270 0.25 0.06 10 0
           10 3 1 0.1 270 270 0.25 0.06 10 0
           10 3 1 1   270 270 0.25 0.06 10 0
           10 3 1 0.1 270 270 0.25 0.06 10 0
           10 3 1 1   270 270 0.25 0.06 10 0
           10 3 1 0.1 270 270 0.25 0.06 10 0];

[rownum, colnum] = size(profile);    % rownum: # of layers

%%% 1) load Gauss-Legendre nodes and weights
load xk3; load wx3;                  % M = 16
a1 = 0; b1 = 1;                      % Note: cos(theta) from 0 to 1
nodes1 = a1 + (b1-a1)*(-xk+1)/2;     % Gauss-Legendre nodes
weights1 = wx*(b1-a1)/2;             % Gauss-Legendre (Christoffel) weights
angle = acos(nodes1);                % angle is theta in radian
mu = cos(angle);                     % 1 by M vector
mu2 = [mu mu];                       % 1 by 2M vector
weights2 = [weights1 weights1];      % 1 by 2M vector
M = length(weights1); I = eye(2*M);
clear a1 b1 wx xk nodes1

%%% 2) Two boundary conditions for the Whole Stack (WS)
Tsbws = 270;         % (K): temp. of the background layer, or 0th layer
Tcbws = 270;         % (K): temp. of the cosmic background, or (n+1)th layer
epssbws = real(6.5);
epscbws = real(1);
Jnum = 5;
d_up  = zeros(2*M,Jnum);
d_vp  = zeros(2*M,Jnum);
d_upp = zeros(2*M,Jnum);
d_vpp = zeros(2*M,Jnum);
dup  = zeros(2*M,Jnum,rownum);
dvp  = zeros(2*M,Jnum,rownum);
dupp = zeros(2*M,Jnum,rownum);
dvpp = zeros(2*M,Jnum,rownum);
dUp  = zeros(2*M,Jnum,rownum,rownum);
dVp  = zeros(2*M,Jnum,rownum,rownum);
dUpp = zeros(2*M,Jnum,rownum,rownum);
dVpp = zeros(2*M,Jnum,rownum,rownum);

%%% 3) njl: # of the Jacobian layer in the whole stack
```

```
%%%    nms: # of the top of the middle stack in the whole stack
Jacobian = input(sprintf('Proceed Jacobian Procedure, Y = 1 or N = 0:  '));
if Jacobian == 1
    Jacobian = 'Y';
elseif Jacobian == 0
    Jacobian = 'N';
else
    disp('Wrong Choice!!! Please input 1 (Yes) or 0 (No)');
end

if (strcmp(Jacobian,'N'))

[ref,tra,ups,dns,EPS,REF,TRA,USR,LSU]=UMRT_Jac_lvl2_RefAux_v2(profile,epssbws
,epscbws,Tsbws,Tcbws,Jacobian);
elseif (strcmp(Jacobian,'Y'))

[ref,tra,ups,dns,EPS,REF,TRA,USR,LSU,d_ref,d_tra,d_ups,d_dns]=UMRT_Jac_lvl2_R
efAux_v2(profile,epssbws,epscbws,Tsbws,Tcbws,Jacobian);
    for njl = 2:rownum-1
        %%% 4) All properties of the Bottom Stack (BS): +z direction
        rownum_bs = njl-1;
        Tsbbs = Tsbws;
        Tcbbs = profile(rownum_bs,6);
        epssbbs = epssbws;

[REF_bs,USR_bs]=UMRT_Jac_BsTs_Ref('BS',rownum,rownum_bs,epssbbs,Tsbbs,Tcbbs,r
ef,tra,ups,dns,EPS);
        U_bs = USR_bs(:,rownum_bs);
        R_bs = REF_bs(:,:,rownum_bs);

        for nms = njl:rownum-1
            for jnum = 1:Jnum
                dref = d_ref(:,:,jnum,njl);
                dtra = d_tra(:,:,jnum,njl);
                dups = d_ups(:,jnum,njl);
                ddns = d_dns(:,jnum,njl);

                %%% 5) All MS properties: _Ref1: +z and _Ref2: -z

[dRms_dn,dTms_dn,dUms,Rms_dn,Tms_dn,Ums]=UMRT_Jac_MS_Ref1_v2(njl,nms,ref,tra,
ups,dns,dref,dtra,dups,EPS);

[dRms_up,dTms_up,dVms,Rms_up,Tms_up,Vms]=UMRT_Jac_MS_Ref2_v2(njl,nms,ref,tra,
ups,dns,dref,dtra,dups,ddns,EPS);

                %%% 6) All TS properties: _Ref1: +z and _Ref2: -z
                rownum_ts = rownum-nms;
                Tsbts = Tcbws;
                Tcbts = 270;
                epssbts = real(EPS(rownum));

[REF_ts,USR_ts]=UMRT_Jac_BsTs_Ref('TS',rownum,rownum_ts,epssbts,Tsbts,Tcbts,r
ef,tra,ups,dns,EPS);
                V_ts = USR_ts(:,rownum_ts);
                R_ts = REF_ts(:,:,rownum_ts);

[d_u_p,d_v_p,d_u_pp,d_v_pp]=UMRT_Jac_lvl1(U_bs,R_bs,V_ts,R_ts,Ums,Vms,Rms_dn,
Tms_dn,Rms_up,Tms_up,dUms,dVms,dRms_dn,dTms_dn,dRms_up,dTms_up);
```

```
                d_up(:,jnum)  = d_u_p;
                d_vp(:,jnum)  = d_v_p;
                d_upp(:,jnum) = d_u_pp;
                d_vpp(:,jnum) = d_v_pp;
            end
            dup(:,:,nms)  = d_up;
            dvp(:,:,nms)  = d_vp;
            dupp(:,:,nms) = d_upp;
            dvpp(:,:,nms) = d_vpp;
        end
        dUp(:,:,:,njl)  = dup;
        dVp(:,:,:,njl)  = dvp;
        dUpp(:,:,:,njl) = dupp;
        dVpp(:,:,:,njl) = dvpp;
    end
else
    disp('Please input Y or N for processing Jacobian or not!!!')
end
```

```
%%% Program Name: UMRT_Jac_lvl2_RefAux_v2
%%% Description:
%%% 1. This code is for calculating all radiation properties of every single
%%%    layer and multilayer stack, and all derivatives of every single layer
%%% 2. INPUT: a. parameters from profile and boundary conditions
%%%           b. output from UMRT_ML_Jac_func_v2.m
%%%              b.1. ups, dns    : [2M, 1] vectors
%%%                   ref, tra    : [2M, 2M] matrices
%%%              b.2. d_ups, d_dns: [2M, jnum] matrices
%%%                   d_ref, d_tra: [2M, 2M, jnum] matrices
%%%
%%%    OUTPUT: a. ups, dns, and USR : [2M, rownum] matrices
%%%               ref, tra, REF, TRA: [2M, 2M, rownum] matrices
%%%               EPS               : [1, 2M] vector
%%%            b. d_ups, d_dns      : [2M, jnum, rownum] matrices
%%%               d_ref, d_tra are  : [2M, 2M, jnum, rownum] matrices
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [ref,tra,ups,dns,EPS,REF,TRA,USR,LSU,d_ref,d_tra,d_ups,d_dns] =
UMRT_Jac_lvl2_RefAux_v2(profile,eps0,epscb,Tsb,Tcb,Jacobian)
[rownum, ~] = size(profile);          % row number of the whole stack
Jnum = 5;                             % currently, we only deal 5 parameters:
                                       % ks, ka, To, tT, and d

%%% setting cosines and weights for plotting and setting B.C. purposes
load xk3; load wx3;                   % M = 16
a1 = 0; b1 = 1;                       % note: cos(theta) from 0 to 1
nodes1 = a1 + (b1-a1)*(-xk+1)/2;      % Gauss-Legendre nodes
weights1 = wx*(b1-a1)/2;              % Gauss-Legendre (Christoffel) weights
angle = acos(nodes1);                 % angle in radian
mu = nodes1; mu2 = [mu mu];           % mu2 is 1 by 2M vector
weights2 = [weights1 weights1];       % weights2 is 1 by 2M vector
M = length(mu);

%%% I. initiate matrices for storing purpose
%%% a) ordinary matrices
ref = zeros(2*M,2*M,rownum);          % ref. of single layers, symmetric
tra = zeros(2*M,2*M,rownum);          % tra. of single layers, symmetric
REF = zeros(2*M,2*M,rownum);          % total ref. of stack, downwardly: -z
TRA = zeros(2*M,2*M,rownum);          % total tra. of stack, downwardly, -z
LSU = zeros(2*M,2*M,rownum);          % tra. opera. of source, upwardly: +z
ups = zeros(2*M,rownum);              % up. self-rad. of single layers, +z
dns = zeros(2*M,rownum);              % dn. self-rad. of single layers, -z
USR = zeros(2*M,rownum);              % up. self-rad. of stack, +z
EPS = zeros(1,rownum);                % permittivity of each layer

%%% b) derivative matrices
d_ref = zeros(2*M,2*M,Jnum,rownum);   % derivative of ref. of single layers
d_tra = zeros(2*M,2*M,Jnum,rownum);   % derivative of tra. of single layers
d_ups = zeros(2*M,Jnum,rownum);       % derivative of up. self-rad. of SL
d_dns = zeros(2*M,Jnum,rownum);       % derivative of dn. self-rad. of SL

%%% II. compute and store all layer and stack properties
if (strcmp(Jacobian, 'N'))
    if rownum == 1                    % stack contains only 1 layer
        [u1,v1,r1,t1,epsL1] = UMRT_ML_Jac_func_v2(profile);
        ref(:,:,1) = r1;  ups(:,1) = u1;
        tra(:,:,1) = t1;  dns(:,1) = v1;
```

```
        eps1 = real(epsL1);
        [~,T01,R10,T10] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
         Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        [R12,T12,R21,T21] = UMRT_ML_RTU3hi(angle,r1,eps1,epscb);
        [R1,T1,U1,L1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
        EPS(1)     = eps1;
         REF(:,:,1) = R1;
        TRA(:,:,1) = T1;
        LSU(:,:,1) = L1;
         USR(:,1)   = U1;
        Vs_top = (sqrt(weights2.*mu2)*Tcb)';
        Ue = U1 + R1*Vs_top;  % Ue is used for energy conservation check
    else
        [u1 v1 r1 t1 epsL1] = UMRT_ML_Jac_func_v2(profile(1,:));
        ref(:,:,1) = r1;  ups(:,1) = u1;
        tra(:,:,1) = t1;  dns(:,1) = v1;
        eps1 = real(epsL1);
        [~,T01,R10,T10] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        for L = 1:rownum-1
            [u2 v2 r2 t2 epsL2] = UMRT_ML_Jac_func_v2(profile(L+1,:));
            ref(:,:,L+1) = r2;  ups(:,L+1) = u2;
            tra(:,:,L+1) = t2;  dns(:,L+1) = v2;
            eps2 = real(epsL2);
            [R12,T12,R21,T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [R1,T1,U1,L1]  =  UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
            EPS(L)     = eps1;
            REF(:,:,L) = R1;
            TRA(:,:,L) = T1;
            LSU(:,:,L) = L1;
            USR(:,L)   = U1;
            eps1 = eps2; r1 = r2; t1 = t2; u1 = u2; v1 = v2;
            Us = U1; R10 = R1; T10 = T1;
        end
        [R12,T12,R21,T21] = UMRT_ML_RTU3hi(angle,r1,eps1,epscb);
        [R1,T1,U1,L1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
        EPS(rownum)     = eps1;
        REF(:,:,rownum) = R1;
        TRA(:,:,rownum) = T1;
        LSU(:,:,rownum) = L1;
        USR(:,rownum)   = U1;
        Vs_top = (sqrt(weights2.*mu2)*Tcb)';
        Ue = U1 + R1*Vs_top;
    end

elseif (strcmp(Jacobian, 'Y'))
    if rownum == 1
        [u1 v1 r1 t1 epsL1 d_u1 d_v1 d_r1 d_t1] =
UMRT_ML_Jac_func_v2(profile);
        ref(:,:,1) = r1;   d_ref(:,:,:,1) = d_r1;
         tra(:,:,1) = t1;   d_tra(:,:,:,1) = d_t1;
        ups(:,1)   = u1;   d_ups(:,:,1)   = d_u1;
        dns(:,1)   = v1;   d_dns(:,:,1)   = d_v1;
        eps1 = real(epsL1);
        [~,T01,R10,T10] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        [R12,T12,R21,T21] = UMRT_ML_RTU3hi(angle,r1,eps1,epscb);
        [R1,T1,U1,L1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
```

```
        EPS(1)     = eps1;
        REF(:,:,1) = R1;
        TRA(:,:,1) = T1;
        LSU(:,:,1) = L1;
        USR(:,1)   = U1;
        Vs_top = (sqrt(weights2.*mu2)*Tcb)';
        Ue = U1 + R1*Vs_top;
    else
        [u1 v1 r1 t1 epsL1 d_u1 d_v1 d_r1 d_t1] =
UMRT_ML_Jac_func_v2(profile(1,:));
        ref(:,:,1) = r1;   d_ref(:,:,:,1) = d_r1;
        tra(:,:,1) = t1;   d_tra(:,:,:,1) = d_t1;
        ups(:,1) = u1;     d_ups(:,:,1)   = d_u1;
        dns(:,1) = v1;     d_dns(:,:,1)   = d_v1;
        eps1 = real(epsL1);
        [~,T01,R10,T10] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        for L = 1:rownum-1
            [u2 v2 r2 t2 epsL2 d_u2 d_v2 d_r2 d_t2] =
UMRT_ML_Jac_func_v2(profile(L+1,:));
            ref(:,:,L+1) = r2;   d_ref(:,:,:,L+1) = d_r2;
            tra(:,:,L+1) = t2;   d_tra(:,:,:,L+1) = d_t2;
            ups(:,L+1) = u2;     d_ups(:,:,L+1)   = d_u2;
            dns(:,L+1) = v2;     d_dns(:,:,L+1)   = d_v2;
            eps2 = real(epsL2);
            [R12,T12,R21,T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [R1,T1,U1,L1] =  UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
            EPS(L)     = eps1;
            REF(:,:,L) = R1;
            TRA(:,:,L) = T1;
            LSU(:,:,L) = L1;
            USR(:,L)   = U1;
            eps1 = eps2; r1 = r2; t1 = t2; u1 = u2; v1 = v2;
            Us = U1; R10 = R1; T10 = T1;
        end
        [R12,T12,R21,T21] = UMRT_ML_RTU3hi(angle,r1,eps1,epscb);
        [R1,T1,U1,L1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
        EPS(rownum)     = eps1;
        REF(:,:,rownum) = R1;
        TRA(:,:,rownum) = T1;
        LSU(:,:,rownum) = L1;
        USR(:,rownum)   = U1;
        Vs_top = (sqrt(weights2.*mu2)*Tcb)';
        Ue = U1 + R1*Vs_top;
    end
else
end

clear a1 b1 wx xk mu2 weights2
```

```
%%% Program Name: UMRT_Jac_BsTs_Ref
%%% Description:
%%% 1. This code is for calculating (R, U) for the bottom stack (BS, +z) and
%%%    the top stack (TS, -z) based on UMRT_Jac_lvl2_RefAux_v2.m.
%%% 2. Input and output are similar to that of the UMRT_Jac_lvl2_RefAux_v2.m.
%%% 3. Note:
%%%    a. When used for BS, calculated in +z (upward) direction
%%%    a.1. The 1st layer of BS is the 1st (bottom) layer of the whole stack.
%%%    a.2. The top layer of BS is the layer #(njl-1) right below the
%%%         Jacobian layer #njl of the whole stack.
%%%    a.3. eps0 is the permittivity of background
%%%    a.4. epscb is the permittivity of cosmic background
%%%
%%%    b. When used for TS, calculated in -z (downward) direction
%%%    b.1. The 1st layer of TS is the top layer of the whole stack.
%%%    b.2. The top layer of TS is the layer #(nms+1) right above the top of
%%%         middle stack (MS)
%%%    b.3. eps0 is the permittivity of background
%%%    b.4. epscb is the permittivity of cosmic background
%%% 4. The code is last modified by Miao Tian, CET, 07/06/2012.

function [REF_dn,USR_up] =
UMRT_Jac_BsTs_Ref(opt,rownum,rownum1,eps0,Tsb,Tcb,ref,tra,ups,dns,EPS)

%%% setting cosines and weights for plotting and setting B.C. purposes
load xk3; load wx3;                              % M = 16
a1 = 0; b1 = 1;                                  % note: cos(theta) from 0 to 1
nodes1 = a1 + (b1-a1)*(-xk+1)/2;                 % Gauss-Legendre nodes
weights1 = wx*(b1-a1)/2;                         % Gauss-Legendre (Christoffel)
weights
angle = acos(nodes1);                            % angle in radian
mu = nodes1; mu2 = [mu mu];                      % mu2 is 1 by 2M vector
weights2 = [weights1 weights1];                  % weights2 is 1 by 2M vector
M = length(mu);                                  % M = 16
REF_dn = zeros(2*M,2*M,rownum1);                 % downwardly: top to bottom, -z
USR_up = zeros(2*M,rownum1);                     % Upwelling self-radiation, +z

%%% Both the BS and TS don't include the Jacobian layer
%%% BS properties are calculated in +z direction
if (strcmp(opt, 'BS'))
    if rownum1 == 1
        r1 = ref(:,:,1);  u1 = ups(:,1);
        t1 = tra(:,:,1);  v1 = dns(:,1);
        eps1 = EPS(1);
        [~,T01,R10,~] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        [R12 T12 R21 T21] = UMRT_ML_RTU3hi(angle,r1,eps1,eps1);
        [R1_dn,~,U1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
        REF_dn(:,:,rownum1) = R1_dn;
        USR_up(:,rownum1)   = U1;
    else                                         % if BS is multilayer stack
        r1 = ref(:,:,1);  u1 = ups(:,1);
        t1 = tra(:,:,1);  v1 = dns(:,1);
        eps1 = EPS(1);
        [~,T01,R10,~] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        for L = 1:rownum1-1
```

```
            r2 = ref(:,:,L+1);  u2 = ups(:,L+1);
            t2 = tra(:,:,L+1);  v2 = dns(:,L+1);
            eps2 = EPS(L+1);
            [R12 T12 R21 T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [R1_dn,T1_dn,U1] =
UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
            REF_dn(:,:,L) = R1_dn;
            USR_up(:,L)   = U1;
            eps1 = eps2; r1 = r2; t1 = t2; u1 = u2; v1 = v2;
            Us = U1; R10 = R1_dn; T01 = T1_dn;
        end
        [R12 T12 R21 T21] = UMRT_ML_RTU3hi(angle,r1,eps1,eps1);
        [R1_dn,~,U1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
        REF_dn(:,:,rownum1) = R1_dn;
        USR_up(:,rownum1)   = U1;
    End
%%% TS properties are calculated in -z direction
%%% note: r,t matrices are symmetric, however u and v need to be swapped
%%%       since now the direction is -z.
elseif (strcmp(opt, 'TS'))
    if rownum1 == 1
        r1 = ref(:,:,rownum);  v1 = ups(:,rownum);
        t1 = tra(:,:,rownum);  u1 = dns(:,rownum);
        eps1 = EPS(rownum);
        [~,T01,R10,~] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        [R12 T12 R21 T21] = UMRT_ML_RTU3hi(angle,r1,eps1,eps1);
        [R1_dn,~,U1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
        REF_dn(:,:,rownum1) = R1_dn;
        USR_up(:,rownum1)   = U1;
    else                                          % if TS is multilayer stack
        r1 = ref(:,:,rownum);  v1 = ups(:,rownum);
        t1 = tra(:,:,rownum);  u1 = dns(:,rownum);
        eps1 = EPS(rownum);
        [~,T01,R10,~] = UMRT_ML_RTU3ih(angle,r1,eps0,eps1);
        Us = T01*(sqrt(weights2.*mu2)*Tsb)';
        for L = 1:rownum1-1
            r2 = ref(:,:,rownum-L);  v2 = ups(:,rownum-L);
            t2 = tra(:,:,rownum-L);  u2 = dns(:,rownum-L);
            eps2 = EPS(rownum-L);
            [R12 T12 R21 T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [R1_dn,T1_dn,U1] =
UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
            REF_dn(:,:,L) = R1_dn;
            USR_up(:,L)   = U1;
            eps1 = eps2; r1 = r2; t1 = t2; u1 = u2; v1 = v2;
            Us = U1; R10 = R1_dn; T01 = T1_dn;
        end
        [R12 T12 R21 T21] = UMRT_ML_RTU3hi(angle,r1,eps1,eps1);
        [R1_dn,~,U1] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T01);
        REF_dn(:,:,rownum1) = R1_dn;
        USR_up(:,rownum1)   = U1;
    end
else
    disp('UMRT_Jac_lvl2_RefAuxBsTs: please input BS or TS as opt!!!')
end
```

```
%%% Program Name: UMRT_Jac_MS_Ref1_v2
%%% Description:
%%% 1. This code is for calculating all radiation and derivative quantities
%%%    of single and multi- layer in the middle stack (MS): UPWARDLY, +z
%%%
%%% 2. INPUT:  a. njl: # of the Jacobian layer
%%%               nms: # of the observation layer
%%%            b. ups, dns  : [2M, rownum] matrices
%%%               ref, tra  : [2M, 2M, rownum] matrices
%%%               dups, ddns: [2M, 1] vectors,   w.r.t to a (jnum, njl)
%%%               dref, dtra: [2M, 2M] matrices, w.r.t to a (jnum, njl)
%%%    OUTPUT: a. R1, T1 and dR1, dT1: [2M, 2M] matrices
%%%               U1 and dU1         : [2M, 1] vectors
%%% 3. The code is last modified by Miao Tian, CET, 07/06/2012.

function [dR1,dT1,dU1,R1,T1,U1] =
UMRT_Jac_MS_Ref1_v2(njl,nms,ref,tra,ups,dns,dref,dtra,dups,EPS)

load xk3; load wx3;                                % M = 16
a1 = 0; b1 = 1;                                    % note: cos(theta) from 0 to 1
nodes1 = a1 + (b1-a1)*(-xk+1)/2;                   % Gauss-Legendre nodes
angle = acos(nodes1);                              % angle in radian

if njl > nms
    disp('WARNING: the Jacobians are found above the perturbed layer (njl <
nms)!!!');
    return;
elseif njl == nms                                  % MS contains only 1 layer
    R1 = ref(:,:,njl);  dR1 = dref;
    T1 = tra(:,:,njl);  dT1 = dtra;
    U1 = ups(:,njl);    dU1 = dups;
else                                               % MS is multilayer stack
    r1 = ref(:,:,njl);  dr1 = dref;
    t1 = tra(:,:,njl);  dt1 = dtra;
    u1 = ups(:,njl);    du1 = dups;
    r2 = ref(:,:,njl+1);
    eps1 = real(EPS(njl));
    eps2 = real(EPS(njl+1));
    [~,dT12,dR21,dT21,~,T12,R21,T21] =
UMRT_ML_RTU3iim_Diff1(angle,r1,r2,dr1,eps1,eps2);

%%% Layer #1 is the Jacobian layer, and layer #0 (background) is neutral.
%%% 1) R10_tet = 0, R01_tet = r1, T10_tet = T01_tet = I. (A.K.Fung, p.351)
%%% 2) R1_dn = R21_hat, T1_dn = t1*T21_hat.              (A.K.Fung, p.358)
%%% 3) Lu1 = T12_hat, Ld1 = 0, Ll1 = T12_hat*t1          (A.K.Fung, p.367)
%%% The total R,T,U at the TOP surface of LAYER #1 are

    R1 = R21;     dR1 = dR21;                      % Downwelling, -z
    T1 = t1*T21;  dT1 = dt1*T21 + t1*dT21;         % Downwelling, -z
    U1 = T12*u1;  dU1 = dT12*u1 + T12*du1;         % No radiation from Layer #0
    R10 = R1; dR10 = dR1;
    T10 = T1; dT10 = dT1;
    Us  = U1; dUs  = dU1;
    eps1 = eps2;

    if njl == (nms-1)                              % MS contains only 2 layers
```

```
    else                                          % MS: >= 3 layers (nms >= 3)
        for i = njl+1:nms-1                       % from #(njl+1) to #(nms-1)
            r1 = ref(:,:,i);  u1 = ups(:,i);
            t1 = tra(:,:,i);  v1 = dns(:,i);
            r2 = ref(:,:,i+1);
            eps2 = real(EPS(i+1));
            [R12 T12 R21 T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [dR1,dT1,dU1,R1,T1,U1] =
UMRT_ML_RTU4_Diff1(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10,dUs,dR10,dT10);
            R10 = R1; dR10 = dR1;
            T10 = T1; dT10 = dT1;
            Us  = U1; dUs  = dU1;
            eps1 = eps2;
        end
    end

    r1 = ref(:,:,nms);  u1 = ups(:,nms);
    t1 = tra(:,:,nms);  v1 = dns(:,nms);
    [R12 T12 R21 T21] = UMRT_ML_RTU3hi(angle,r1,eps1,eps1);
    [dR1,dT1,dU1,R1,T1,U1] =
UMRT_ML_RTU4_Diff1(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10,dUs,dR10,dT10);
end
```

```
%%% Program Name: UMRT_Jac_MS_Ref2_v2
%%% Description: This code is the counterpart of UMRT_Jac_MS_Ref1_v2. The
%%%              only difference is that it computes DOWNWARDLY, -z.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [dR1,dT1,dU1,R1,T1,U1] =
UMRT_Jac_MS_Ref2_v2(njl,nms,ref,tra,ups,dns,dref,dtra,dups,ddns,EPS)

load xk3; load wx3;                              % M = 16
a1 = 0; b1 = 1;                                  % note: cos(theta) from 0 to 1
nodes1 = a1 + (b1-a1)*(-xk+1)/2;                 % Gauss-Legendre nodes
angle = acos(nodes1);                            % angle in radian
M = length(angle); I = eye(2*M);

if njl > nms
    disp('WARNING: the Jacobians are found above the perturbed layer (njl <
nms)!!!');
    return;
elseif njl == nms                                % MS contains only 1 layer
    R1 = ref(:,:,njl);  dR1 = dref;
    T1 = tra(:,:,njl);  dT1 = dtra;
    U1 = dns(:,njl);    dU1 = ddns;
else
    R10 = zeros(2*M,2*M); T10 = I; Us = zeros(2*M,1);
    r1 = ref(:,:,nms); t1 = tra(:,:,nms); u1 = dns(:,nms); v1 = ups(:,nms);
    eps1 = real(EPS(nms));

    if njl == (nms-1)                            % MS contains only 2 layers
        r2 = ref(:,:,njl); dr2 = dref;
        t2 = tra(:,:,njl); dt2 = dtra;
        u2 = dns(:,njl);   du2 = ddns;
        v2 = ups(:,njl);   dv2 = dups;
        eps2 = real(EPS(njl));
        [dR12,dT12,dR21,dT21,R12,T12,R21,T21] =
UMRT_ML_RTU3iim_Diff2(angle,r1,r2,dr2,eps1,eps2);
        [dR2,dT2,dU2,R2,T2,U2] =
UMRT_ML_RTU4_Diff2(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10,dR12,dT12,dR21,dT21);
    else                                         % MS: >= 3 layers (nms >= 3)
        for i = nms-1:-1:njl+1
            r2 = ref(:,:,i);  v2 = ups(:,i);
            t2 = tra(:,:,i);  u2 = dns(:,i);
            eps2 = real(EPS(i));
            [R12,T12,R21,T21] = UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2);
            [R2,T2,U2] = UMRT_ML_RTU4(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10);
            R10 = R2; T10 = T2; Us = U2;
            r1 = r2; t1 = t2; u1 = u2; v1 = v2; eps1 = eps2;
        end

        r2 = ref(:,:,njl); dr2 = dref;
        t2 = tra(:,:,njl); dt2 = dtra;
        u2 = dns(:,njl);   du2 = ddns;
        v2 = ups(:,njl);   dv2 = dups;

        [dR12,dT12,dR21,dT21,R12,T12,R21,T21] =
UMRT_ML_RTU3iim_Diff2(angle,r1,r2,dr2,eps1,eps2);
        [dR2,dT2,dU2,R2,T2,U2] =
```

```
UMRT_ML_RTU4_Diff2(t1,u1,v1,Us,T12,T21,R12,R21,R10,T10,dR12,dT12,dR21,dT21);
    end

    R1 = t2*R2*t2;
    T1 = T2*t2;
    U1 = u2 + t2*R2*v2 + t2*U2;
    dR1 = (dt2*R2 + t2*dR2)*t2 + t2*R2*dt2;
    dT1 = dT2*t2 + T2*dt2;
    dU1 = du2 + (dt2*R2 + t2*dR2)*v2 + t2*R2*dv2 + dt2*U2 + t2*dU2;
end
```

```
%%% Program Name: UMRT_Jac_lvl1
%%% Description: this code is for calculating FOUR derivative quantities.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [d_u_p d_v_p d_u_pp d_v_pp] =
UMRT_Jac_lvl1(U,R1,V,R2,Us,Vs,Rs_dn,Ts_dn,Rs_up,Ts_up,dUs,dVs,dRs_dn,dTs_dn,d
Rs_up,dTs_up)

I = eye(length(R1));

a11 = I - Rs_dn * R2; a12 = - Ts_up * R1;
a21 = - Ts_dn * R2;   a22 = I - Rs_up * R1;

b1 = Rs_dn * (V + R2 * Us) + Ts_up * (U + R1 * Vs);
b2 = Ts_dn * (V + R2 * Us) + Rs_up * (U + R1 * Vs);

d_a11 = - dRs_dn * R2;  d_a12 = - dTs_up * R1;
d_a21 = - dTs_dn * R2;  d_a22 = - dRs_up * R1;

d_b1a = dRs_dn * (V + R2 * Us) + Rs_dn * R2 * dUs;
d_b1b = dTs_up * (U + R1 * Vs) + Ts_up * R1 * dVs;
d_b1  = d_b1a + d_b1b;

d_b2a = dTs_dn * (V + R2 * Us) + Ts_dn * R2 * dUs;
d_b2b = dRs_up * (U + R1 * Vs) + Rs_up * R1 * dVs;
d_b2  = d_b2a + d_b2b;

u_pp = (a11 - a12 / a22 * a21) \ (b1 - a12 / a22 * b2);
v_p  = (a22 - a21 / a11 * a12) \ (b2 - a21 / a11 * b1);
u_p  = R1 * (Vs + v_p);
v_pp = R2 * (Us + u_pp);

c1 = d_b1 - d_a11*u_pp - d_a12*v_p;
c2 = d_b2 - d_a21*u_pp - d_a22*v_p;

d_u_pp = (a11 - a12 / a22 * a21) \ (c1 - a12 / a22 * c2);
d_v_p  = (a22 - a21 / a11 * a12) \ (c2 - a21 / a11 * c1);
d_u_p  =  R1 * (dVs + d_v_p);
d_v_pp = R2 * (dUs + d_u_pp);
```

```
%%% Program Name: UMRT_ML_Jac_func_v2
%%% Description:
%%% 1. This code is the core of UMRT-Jacobian. It computes all radiation and
%%%    derivative quantities for each every single layer
%%% 2. INPUT:  a. profile: n by m matrix, where n is the number of layers
%%%                        used for the multilayer model and m is the number
%%%                        of the parameters needed for each individual layer
%%%    OUTPUT: a. ups, dns are: [2M, 1] vectors
%%%               ref, tra are: [2M, 2M] matrices
%%%            b. d_ups, d_dns are: [2M, jnum] matrices
%%%               d_ref, d_tra are: [2M, 2M, jnum] matrices
%%%
%%% 3. Note:   a. dA, dB are [2M, 2M, jnum] matrices.
%%%            b. dAt, dBt are extracted from dA and dB w.r.t ONE parameter
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [ups dns ref tra eps d_ups d_dns d_ref d_tra] =
UMRT_ML_Jac_func_v2(profile)
freq        = profile(1);  % frequency in GHz
typeopt     = profile(2);  % phase matrix
materialopt = profile(3);  % water or ice or ... of the lth layer
thickness   = profile(4);  % thickness of the lth layer
Tbot        = profile(5);  % physical temperature at top of the lth layer
Ttop        = profile(6);  % physical temperature at top of the lth layer
Fv          = profile(7);  % fractional volume [0 1]
Dia         = profile(8);  % mean diameter, <D> (cm)
PR          = profile(9);  % precipitation rate (mm/hr)

%%% Choosing type of phase matrix
if typeopt == 1
    type = 'Mie';
elseif typeopt == 2
    type = 'DMRT';
elseif typeopt == 3
    type = 'HG';
elseif typeopt == 4
    type = 'Rayleigh';
else
    disp('UMRT_SL_DMRTver2.m: Wrong Phase Matrix Choice!');
end

%%% Choosing material
if materialopt == 1
    material = 'water';
elseif materialopt == 2
    material = 'ice';
else
    disp('UMRT_SL_DMRTver2.m: Wrong Material Choice!');
end

%%% Choosing type of temperature profile
slope = (Ttop - Tbot) / thickness;
if slope == 0
    radtype = 'constant';
else
    radtype = 'linear';
end
```

```
%%% used for computing permittivity
Tdiel = (Tbot + Ttop)/2;
clear typeopt materialopt profopt

%%%%%%%%%%%%%%%%%%%%%% Calculate A, B, dA, dB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ka,ks,weights1,angle,mu,eps,A,B,dA,dB] =
UMRT_Jac_lvl7_v2(freq,type,material,Fv,Dia,PR,Tdiel);
M = length(angle);
d_ref = zeros(2*M,2*M,5);   % New_06/26/2012: can handle 5 parameters now.
d_tra = zeros(2*M,2*M,5);
d_ups = zeros(2*M,5);
d_dns = zeros(2*M,5);

for jnum = 1:5              %%% New_06/26/2012: can handle 5 parameters now.
    if jnum == 0            %%% Choosing partial derivative parameter
        ivar = 'none';
    elseif jnum == 1
        ivar = 'ks';
    elseif jnum == 2
        ivar = 'ka';
    elseif jnum == 3
        ivar = 'Tbot';
    elseif jnum == 4
        ivar = 'slope';
    elseif jnum == 5
        ivar = 'thickness';
    else
        disp('UMRT_SL_DMRTver2.m: This parameter is not included yet!');
    end
    dAt = dA(:,:,jnum);
    dBt = dB(:,:,jnum);
    [M1,M2,dM1,dM2,Lam1,Lam2,dLam1,dLam2] = UMRT_Jac_lvl6(A,B,dAt,dBt,type);
    [d_zeta1,d_a1,d_bt1,dbr,zeta,a,bt,br,dM1t] =
UMRT_Jac_lvl5(M1,M2,dM1,dM2,Lam1,Lam2,dLam1,dLam2,thickness,ivar);
    [ref,tra,dref,dtra] =
UMRT_Jac_lvl4(M1,dM1,dM1t,d_zeta1,d_a1,d_bt1,dbr,zeta,a,bt,br,ivar);
    [ups,dns,dups,ddns] =
UMRT_Jac_lvl3_v2(A,B,ref,tra,dAt,dBt,dref,dtra,weights1,mu,ka,Tbot,Ttop,thick
ness,radtype,1,jnum);
    d_ups(:,jnum) = dups;   d_ref(:,:,jnum) = dref;
    d_dns(:,jnum) = ddns;   d_tra(:,:,jnum) = dtra;
end
end
```

```
%%% Program Name: UMRT_Jac_lvl7_v2
%%% Description:
%%% 1. This code is mainly for computing the two MOST important matrices
%%%    A and B along with their derivatives dA and dB w.r.t to (currently)
%%%    the five parameters
%%% 2. INPUT:  a. parameters extracted from profile
%%%    OUTPUT: a. A, B:   [2M, 2M] matrices
%%%            b. dA, dB: [2M, 2M, jnum] matrices
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [ka ks weights1 angle mu eps A B dA dB] =
UMRT_Jac_lvl7_v2(freq,type,material,Fv,Dia,PR,Tdiel)

Jnum = 5;       % currently, we only deal 5 parameters: ks, ka, To, tT, and d
%%% Original unit:  Mie: naper/m and DMRT: naper/cm
%%% This code uses: Mie-Sparse: naper/km (x1000); Mie-Dense: naper/m (x1)
%%%                 DMRT: naper/m (x100)
if(strcmp(material,'ice'))
    factor = 1.028*1.028;   % ice-to-water volume change, not applied in DMRT
    if(strcmp(type,'DMRT'))
        scale = 1;% NEW_06/26/2012 no longer convert from naper/cm to naper/m
        [Ks,Ka,epsilon_p,K_eff,n_max,no,k0fs,k0a,kfs,k] =
DMRTKs_UMRT(freq,Tdiel,Fv,Dia);
        eps = (real(K_eff)/kfs)^2;
        ka  = Ka*scale; ks = Ks*scale; ke = ks + ka;
    elseif(strcmp(type,'Mie')||strcmp(type,'Rayleigh')||strcmp(type,'HG'))
            eps    = conj(h2o_ice_diel(freq, Tdiel));
            option = input('Enter sparse (SS) or dense (Gamma, 25%) for Mie-
Ice: ');
        if(strcmp(option,'sparse'))
            SDFopt = 8;               % SS SDF for sparse ice particles
            scale  = 1;
            lab    = 2.29*PR^(-0.45); % lambda, mm^-1
            mD     = 1/lab;           % mean diameter, mm
            [kg,Ks,Ka]=PolyMieCoeffWaterIce(freq,Tdiel,mD,PR,SDFopt,material);
            ka = (Ka*factor)*scale; ks = (Ks*factor)*scale; ke = ks + ka;
        elseif(strcmp(option,'dense'))
            SDFopt = 9;               % Gamma SDF for dense ice particles
            scale  = 1;               % Dense Mie: Naper/m to Naper/m
            mD     = Dia;
            [kg,Ks,Ka]=PolyMieCoeffWaterIce(freq,Tdiel,mD,PR,SDFopt,material);
            ka = (Ka*factor)*scale; ks = (Ks*factor)*scale; ke = ks + ka;
        else
            disp('UMRT_SubMatrices.m: Wrong choice of option!!');
        end
    else
        disp('UMRT_SubMatrices.m: Wrong phase matrix type for ice');
        return;
    end

elseif(strcmp(material,'water'))
    if(strcmp(type,'Rayleigh')||strcmp(type,'HG'))
        scale  = 1;                   % NEW_06/26/2012 no longer convert from
naper/m to naper/km
        eps    = conj(d3lec(freq,Tdiel,0,2));
        SDFopt = 2;                   % MP SDF for rain
        lab    = 4.1*PR^(-0.21);      % lambda, mm^-1
```

```
        mD     = 1/lab;               % mean diameter, mm
        [kg,Ks,Ka]=PolyMieCoeffWaterIce(freq,Tdiel,mD,PR,SDFopt,material);
        ka = Ka*scale; ks = Ks*scale;
        ke = ka + ks;
    elseif (strcmp(type,'Mie'))
        scale  = 1;
        eps    = conj(d3lec(freq,Tdiel,0,2));
        SDFopt = 2;                   % MP SDF for rain
        lab    = 4.1*PR^(-0.21);      % lambda, mm^-1
        mD     = 1/lab;               % mean diameter, mm
        [~,~,Ka]=PolyMieCoeffWaterIce(freq,Tdiel,mD,PR,SDFopt,material);
        ka = Ka*scale;
        [Ksv,Ksh]=KsMieMatrix(freq,eps,mD,SDFopt,PR,scale);
        ks = (mean(Ksv+Ksh)/2);
        ke = ka + ks;
    else
        disp('UMRT_SubMatrices: wrong phase matrix type for water/rain!!');
    end
else
    disp('UMRT_SubMatrices.m: wrong material. Enter either water or ice')
    return;
end

%%%%%%%%%%%%%%%%%%%% End of Setting General Parameters %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% load Gauss-Legendre nodes and weights
load xk3; load wx3;          % M = 16
a1 = 0; b1 = 1;              % Note: cos(theta) from 0 to 1
nodes1   = a1 + (b1-a1)*(-xk+1)/2;
weights1 = wx*(b1-a1)/2;
angle    = acos(nodes1);     % angle is theta in radian
the_pn   = pi-angle;         % the_pn is in radian
mu       = cos(angle);       % mu is cos(theta), eqn (5)
M        = length(angle);
Ao  = zeros(M,M); Bo  = zeros(M,M); Co  = zeros(M,M); Do  = zeros(M,M);
Eo  = zeros(M,M); Fo  = zeros(M,M); Go  = zeros(M,M); Ho  = zeros(M,M);
%%% All temporary derivative matrices are 3-D, the 3rd number is related to
%%% the interested parameters
dAo = zeros(M,M,Jnum); dBo = zeros(M,M,Jnum); dCo = zeros(M,M,Jnum);
dDo = zeros(M,M,Jnum); dEo = zeros(M,M,Jnum); dFo = zeros(M,M,Jnum);
dGo = zeros(M,M,Jnum); dHo = zeros(M,M,Jnum);
clear a1 b1 wx xk

%%%%%%%%%%%%%%%%%%%%%%%%% 1) Set Forward Matrices %%%%%%%%%%%%%%%%%%%%%%%%%
%%% 2) Differentiate Forward Matrices With Respect To (W.R.T.) ks or ka %%%
 ke_diag = ke./mu;
dke_diag = 1./mu;                % differentiate ke_diag w.r.t ks or ka.
matlabpool open
if (strcmp(type,'Mie'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_s)
        ksv2 = Ksv(i)*1e6/scale;
        ksh2 = Ksh(i)*1e6/scale;
        parfor j = 1:M
            theta_i = angle(j);  % theta_i is in radian (0 to pi/2)
```

```
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
[nMiePM]=Mie_NRPM(theta_s,theta_i,freq,eps,mD,SDFopt,PR,ksv2,ksh2);
            Ao(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(1,1);
            Co(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(2,1);
            Eo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(1,2);
            Go(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(2,2);
            %%% New_06/26/2012: since dA, dB are zero when differentiated to
            %%% To, tT and d, nothing changed for all phase matrices.
            dAo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(1,1);
            dCo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(2,1);
            dEo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(1,2);
            dGo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(2,2);
        end
    end
elseif (strcmp(type,'HG'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_s)
        parfor j = 1:M
            theta_i = angle(j);  % theta_i is in radian (0 to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
           [HGave]=HGPM(theta_s,theta_i,kg);            % referred to radian
           Ao(i,j)  = -ks*sqrt(gi*gj/mui/muj)*HGave(1,1);
           Co(i,j)  = 0; Eo(i,j) = 0;
           Go(i,j)  = -ks*sqrt(gi*gj/mui/muj)*HGave(2,2);
           dAo(i,j,1) = Ao(i,j)/ks;
           dGo(i,j,1) = Go(i,j)/ks;
        end
    end
elseif (strcmp(type,'Rayleigh'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_s)
        parfor j = 1:M
            theta_i = angle(j);  % theta_i is in radian (0 to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
           [RH] = Rayleigh(theta_s,theta_i);       % referred to radian
           Ao(i,j)  = -ks*sqrt(gi*gj/mui/muj)*RH(1,1);
           Co(i,j)  = 0; Eo(i,j) = 0;
           Go(i,j)  = -ks*sqrt(gi*gj/mui/muj)*RH(2,2);
           dAo(i,j,1) = Ao(i,j)/ks;
           dGo(i,j,1) = Go(i,j)/ks;
        end
    end
elseif (strcmp(type,'DMRT'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_s)
        parfor j = 1:M
            theta_i = angle(j);  % theta_i is in radian (0 to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
```

```
            [DMRTnrpm] =
DMRT_NRPM(freq,epsilon_p,Fv,k0fs,k0a,kfs,k,n_max,no,theta_s,theta_i,Dia,Ks);
            Ao(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(1,1);
            Co(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(2,1);
            Eo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(1,2);
            Go(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(2,2);
            dAo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(1,1);
            dCo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(2,1);
            dEo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(1,2);
            dGo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(2,2);
        end
    end
else
    disp('UMRT_SubMatrices.m: Wrong phase matrix in Forward computation.');
    return;
end

Ao = diag(ke_diag) + Ao;   Go = diag(ke_diag) + Go;
dAo(:,:,1) = diag(dke_diag) + dAo(:,:,1);  % 1: ks
dGo(:,:,1) = diag(dke_diag) + dGo(:,:,1);
dAo(:,:,2) = diag(dke_diag) + dAo(:,:,2);  % 2: ka
dGo(:,:,2) = diag(dke_diag) + dGo(:,:,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% 3) Set Backward Matrices %%%%%%%%%%%%%%%%%%%%%%%%%%
if (strcmp(type,'Mie'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_i)
        ksv2 = Ksv(i)*1e6/scale;
        ksh2 = Ksh(i)*1e6/scale;
        parfor j = 1:M
            theta_i = the_pn(j); % the_pn is in radian (pi to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
[nMiePM]=Mie_NRPM(theta_s,theta_i,freq,eps,mD,SDFopt,PR,ksv2,ksh2);
            Bo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(1,1);
            Do(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(2,1);
            Fo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(1,2);
            Ho(i,j)  = -ks*sqrt(gi*gj/mui/muj)*nMiePM(2,2);
            dBo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(1,1);
            dDo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(2,1);
            dFo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(1,2);
            dHo(i,j,1) = -sqrt(gi*gj/mui/muj)*nMiePM(2,2);
        end
    end
elseif (strcmp(type,'HG'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_i)
        parfor j = 1:M
            theta_i = the_pn(j); % the_pn is in radian (pi to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
            [HGave]=HGPM(theta_s,theta_i,kg);          % referred to radian
```

```
            Bo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*HGave(1,1);
            Do(i,j)  = 0; Fo(i,j) = 0;
            Ho(i,j)  = -ks*sqrt(gi*gj/mui/muj)*HGave(2,2);
            dBo(i,j,1) = Bo(i,j)/ks;
            dHo(i,j,1) = Ho(i,j)/ks;
        end
    end
elseif (strcmp(type,'Rayleigh'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_i)
        parfor j = 1:M
            theta_i = the_pn(j); % the_pn is in radian (pi to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
            [RH] = Rayleigh(theta_s,theta_i);           % referred to radian
            Bo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*RH(1,1);
            Do(i,j)  = 0; Fo(i,j) = 0;
            Ho(i,j)  = -ks*sqrt(gi*gj/mui/muj)*RH(2,2);
            dBo(i,j,1) = Bo(i,j)/ks;
            dHo(i,j,1) = Ho(i,j)/ks;
        end
    end
elseif (strcmp(type,'DMRT'))
    for i = 1:M
        theta_s = angle(i);      % theta_s is in radian (0 to pi/2)
        gi = weights1(i);        % Christoffel weights
        mui = mu(i);             % cos(theta_i)
        parfor j = 1:M
            theta_i = the_pn(j); % the_pn is in radian (pi to pi/2)
            gj = weights1(j);    % Christoffel weights
            muj = mu(j);         % cos(theta_i)
[DMRTnrpm]=DMRT_NRPM(freq,epsilon_p,Fv,k0fs,k0a,kfs,k,n_max,no,theta_s,theta_
i,Dia,Ks);
            Bo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(1,1);
            Do(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(2,1);
            Fo(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(1,2);
            Ho(i,j)  = -ks*sqrt(gi*gj/mui/muj)*DMRTnrpm(2,2);
            dBo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(1,1);
            dDo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(2,1);
            dFo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(1,2);
            dHo(i,j,1) = -sqrt(gi*gj/mui/muj)*DMRTnrpm(2,2);
        end
    end
else
    disp('UMRT_SubMatrices.m: Wrong phase matrix in Backward computation');
    return;
end
Fpm = [Ao Co; Eo Go];      % 2M by 2M matrix
Bpm = [Bo Do; Fo Ho];      % 2M by 2M matrix
A = Fpm + Bpm;             % 2M by 2M matrix, A = Ao+Bo;
B = Fpm - Bpm;             % 2M by 2M matrix, B = Ao-Bo;
dFpm = [dAo dCo; dEo dGo]; % 2M by 2M by 4 matrix
dBpm = [dBo dDo; dFo dHo]; % 2M by 2M by 4 matrix
dA = dFpm + dBpm;          % 2M by 2M by 4 matrix, dA = dAo+dBo;
dB = dFpm - dBpm;          % 2M by 2M by 4 matrix, dB = dAo-dBo;
matlabpool close
```

```
%%% Program Name: UMRT_Jac_lvl6
%%% Description: this code is for calculating the derivative quantities of
%%%              the eigenvector and eigenvalue matrices
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [M1 M2 dM1 dM2 Lam1 Lam2 dLam1 dLam2]=UMRT_Jac_lvl6(A,B,dA,dB,type)

if (strcmp(type,'Rayleigh') || strcmp(type,'HG'))
    s = size(A); leng = s(1);  % the size/length of the matrices: 2M by 2M
    M = leng/2;

    %%% Separate the Pvv and Phh submatrices (M by M), since Pvh = Phv = 0.
    Avv = A(1:M,1:M); Ahh = A(M+1:2*M,M+1:2*M);
    Bvv = B(1:M,1:M); Bhh = B(M+1:2*M,M+1:2*M);

    dAvv = dA(1:M,1:M); dAhh = dA(M+1:2*M,M+1:2*M);
    dBvv = dB(1:M,1:M); dBhh = dB(M+1:2*M,M+1:2*M);

    [M1v M2v dM1v dM2v Lam1v Lam2v dLam1v dLam2v] =
UMRT_Jac_lvl6_Core(Avv,Bvv,dAvv,dBvv);
    [M1h M2h dM1h dM2h Lam1h Lam2h dLam1h dLam2h] =
UMRT_Jac_lvl6_Core(Ahh,Bhh,dAhh,dBhh);

    %%% Form 2M by 2M corresponding matrices
     M1 = zeros(s);  M2 = zeros(s);  Lam1 = zeros(s);  Lam2 = zeros(s);
    dM1 = zeros(s); dM2 = zeros(s); dLam1 = zeros(s); dLam2 = zeros(s);

     M1(1:M,1:M) =  M1v;  M1(M+1:2*M,M+1:2*M) =  M1h;
    dM1(1:M,1:M) = dM1v; dM1(M+1:2*M,M+1:2*M) = dM1h;

     Lam1(1:M,1:M) =  Lam1v;  Lam1(M+1:2*M,M+1:2*M) =  Lam1h;
    dLam1(1:M,1:M) = dLam1v; dLam1(M+1:2*M,M+1:2*M) = dLam1h;

     M2(1:M,1:M) =  M2v;  M2(M+1:2*M,M+1:2*M) =  M2h;
    dM2(1:M,1:M) = dM2v; dM2(M+1:2*M,M+1:2*M) = dM2h;

     Lam2(1:M,1:M) =  Lam2v;  Lam2(M+1:2*M,M+1:2*M) =  Lam2h;
    dLam2(1:M,1:M) = dLam2v; dLam2(M+1:2*M,M+1:2*M) = dLam2h;

elseif (strcmp(type,'Mie') || strcmp(type,'DMRT'))
    [M1 M2 dM1 dM2 Lam1 Lam2 dLam1 dLam2]=UMRT_Jac_lvl6_Core(A,B,dA,dB);
else
    disp('UMRT_Jac_lvl6.m: Wrong Phase Matrix Choice!');
end
```

```
%%% Program Name: UMRT_Jac_lvl6_Core
%%% Description: this code is for calculating the differentiated matrices of
%%%              M1, M2, Lam1, Lam2, M1', M2', Lam1^0.5, Lam2^0.5, Lam1^-0.5,
%%%              Lam2^-0.5
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [M1 M2 dM1 dM2 Lam1 Lam2 dLam1 dLam2]=UMRT_Jac_lvl6_Core(A,B,dA,dB)

s = size(A); leng = s(1);            % find the size/length of the matrices
I = eye(leng);
[M1 Lam1] = eig(A);                  % get eigen -vectors and -values of A
diLam1 = diag(Lam1);                 % vector: eigenvalues of A
                                     % Lam1; diagonal matrix
M1_temp = M1' * dA * M1;             % make a transition matrix of dA
dLam1 = diag(diag(M1_temp));         % matrix: differentiation of lambda 1

dM1 = zeros(s);
for n = 1:leng
    for m = 1:leng
        for i = 1:leng
            if i ~= m                        % index i is not equal to m
                p1 = M1(n,i) * M1_temp(i,m);
                p2 = diLam1(m) - diLam1(i);
                dM1(n,m) = dM1(n,m) + p1/p2; % differentiation of M1
            else
            end
        end
    end
end

dM1t = - (I/M1) * dM1 * M1';             % differentiation of M1_transpose
 Lam1_p12 = Lam1^( 0.5);                 % Lam1^ 0.5; p: positive
dLam1_p12 =  0.5 * Lam1^(-0.5) * dLam1;  % differentiation of Lam1^ 0.5

B2 = Lam1_p12 * M1' * B * M1 * Lam1_p12; % DOTLRT: eqn. (46)
[M2 Lam2] = eig(B2);                     % eigen -vectors and -values of B2

%%% differentiation of B2: differentiation by parts
p3 = dLam1_p12 *  M1' *  B *  M1 *  Lam1_p12;
p4 =  Lam1_p12 * dM1t *  B *  M1 *  Lam1_p12;
p5 =  Lam1_p12 *  M1' * dB *  M1 *  Lam1_p12;
p6 =  Lam1_p12 *  M1' *  B * dM1 *  Lam1_p12;
p7 =  Lam1_p12 *  M1' *  B *  M1 * dLam1_p12;
dB2 = p3 + p4 + p5 + p6 + p7;

M2_temp = M2' * dB2 * M2;                % make a transition matrix of dB2
dLam2 = diag(diag(M2_temp));             % differentiation of lambda 2
diLam2 = diag(Lam2);                     % make eigenvalues in a vector

dM2 = zeros(s);
for n = 1:leng
    for m = 1:leng
        for i = 1:leng
            if i ~= m                    % index i is not equal to m
                p8 = M2(n,i) * M2_temp(i,m);
                p9 = diLam2(m) - diLam2(i);
```

```
                dM2(n,m) = dM2(n,m) + p8/p9; % differentiation of M2
            else
            end
        end
    end
end
```

```
%%% Program Name: UMRT_Jac_lvl6_Aux
%%% Description: this code is for calculating the differentiated matrices of
%%%              M1^-1, M2^-1, Lam1^0.5, Lam2^0.5, Lam1^-0.5, Lam2^-0.5
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [dM1t dM2t Lam1_p12 Lam1_n12 dLam1_p12 dLam1_n12 Lam2_p12 Lam2_n12
dLam2_p12 dLam2_n12] = UMRT_Jac_lvl6_Aux(M1,M2,dM1,dM2,Lam1,Lam2,dLam1,dLam2)

leng = length(M1);
I = eye(leng);

dM1t = - (I/M1) * dM1 * M1';             % differentiation of M1_transpose
 Lam1_p12 = Lam1^( 0.5);                 % Lam1^ 0.5; p: positive
 Lam1_n12 = Lam1^(-0.5);                 % Lam1^-0.5; n: negative
dLam1_p12 =  0.5 * Lam1^(-0.5) * dLam1;  % differentiation of Lam1^ 0.5
dLam1_n12 = -0.5 * Lam1^(-1.5) * dLam1;  % differentiation of Lam1^-0.5

dM2t = - (I/M2) * dM2 * M2';             % differentiation of M2_transpose
 Lam2_p12 = Lam2^( 0.5);                 % Lam2^ 0.5; p: positive
 Lam2_n12 = Lam2^(-0.5);                 % Lam2^-0.5; n: negative
dLam2_p12 =  0.5 * Lam2^(-0.5) * dLam2;  % differentiation of Lam2^ 0.5
dLam2_n12 = -0.5 * Lam2^(-1.5) * dLam2;  % differentiation of Lam2^-0.5
```

```
%%% Program Name: UMRT_Jac_lvl5
%%% Description: this code is for calculating the differentiated matrices
%%%              related to dref, dtra
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [d_zeta1 d_a1 d_bt1 dbr zeta a bt br dM1t] =
UMRT_Jac_lvl5(M1,M2,dM1,dM2,Lam1,Lam2,dLam1,dLam2,thickness,ivar)
leng = length(M1);
I = eye(leng);
h = thickness;

[dM1t dM2t Lam1_p12 Lam1_n12 dLam1_p12 dLam1_n12 Lam2_p12 Lam2_n12 dLam2_p12
dLam2_n12] = UMRT_Jac_lvl6_Aux(M1,M2,dM1,dM2,Lam1,Lam2,dLam1,dLam2);

if (strcmp(ivar,'ks') || strcmp(ivar,'ka') || strcmp(ivar,'Tbot') ||
strcmp(ivar,'slope'))
    part1 = 0.5*Lam2_p12*h;
    part2 =     Lam2_p12*h;
    %%% ts are diagonal matrices
    %%% 1) differentiation of zeta^-1
    t1 = diag(sinh(diag(part2)));
    t2 = diag(cosh(diag(part2)));

    zeta = Lam2_n12 * t1;
    dzeta_p1 = dLam2_n12 * t1;
    dzeta_p2 =  Lam2_n12 * t2 * dLam2_p12 * h;
    dzeta = dzeta_p1 + dzeta_p2;
    d_zeta1 = - (I/zeta) * dzeta * (I/zeta);

    %%% 2) differentiation of a^-1
    t3 = diag(coth(diag(part1)));
    t4 = diag(csch(diag(part1)));
    dt3 = - t4 * t4 * 0.5 * dLam2_p12 * h; % 0.5 is directly from part1

    a = Lam1_p12 * M2 + Lam1_n12 * M2 * Lam2_p12 * t3;
    a_p1 = dLam1_p12 * M2 + Lam1_p12 * dM2;
    a_p2 = dLam1_n12 *  M2 *  Lam2_p12 *  t3;
    a_p3 =  Lam1_n12 * dM2 *  Lam2_p12 *  t3;
    a_p4 =  Lam1_n12 *  M2 * dLam2_p12 *  t3;
    a_p5 =  Lam1_n12 *  M2 *  Lam2_p12 * dt3;

    da = a_p1 + a_p2 + a_p3 + a_p4 + a_p5;
    d_a1 = - (I/a) * da * (I/a);             % error compared with using
'inv': ~1e-12

    %%% 3) differentiation of bt^-1
    t5 = diag(tanh(diag(part1)));
    t6 = diag(sech(diag(part1)));
    dt5 = 0.5 * t6 * t6 * dLam2_p12 * h;

    bt = M2' * Lam1_p12 + Lam2_p12 * t5 * M2' * Lam1_n12;
    br = M2' * Lam1_p12 - Lam2_p12 * t5 * M2' * Lam1_n12;

    bt_p1 = dM2t * Lam1_p12 + M2' * dLam1_p12;
    bt_p2 = dLam2_p12 *  t5 *  M2' *  Lam1_n12;
    bt_p3 =  Lam2_p12 * dt5 *  M2' *  Lam1_n12;
```

```
    bt_p4 =  Lam2_p12 *  t5 * dM2t *  Lam1_n12;
    bt_p5 =  Lam2_p12 *  t5 *  M2' * dLam1_n12;

    dbt = bt_p1 + bt_p2 + bt_p3 + bt_p4 + bt_p5;
    dbr = bt_p1 - bt_p2 - bt_p3 - bt_p4 - bt_p5;

    d_bt1 = - (I/bt) * dbt * (I/bt);

elseif (strcmp(ivar,'thickness'))
    part1 = 0.5*Lam2_p12*h;
    part2 = Lam2_p12*h;

    %%% 1) differentiation of zeta^-1
    t1 = diag(sinh(diag(part2)));
    t2 = diag(cosh(diag(part2)));
    zeta  = Lam2_n12 * t1;
    dzeta = Lam2_n12 * t2 * Lam2_p12;
    d_zeta1 = - (I/zeta) * dzeta * (I/zeta);

    %%% 2) differentiation of a^-1
    t3 = diag(coth(diag(part1)));
    t4 = diag(csch(diag(part1)));
    dt3 = - t4 * t4 * 0.5 * Lam2_p12; % 0.5 is directly from part1

    a = Lam1_p12 * M2 + Lam1_n12 * M2 * Lam2_p12 * t3;
    da =  Lam1_n12 * M2 * Lam2_p12 * dt3;
    d_a1 = - (I/a) * da * (I/a);    % error compared with using 'inv': ~1e-12

    %%% 3) differentiation of bt^-1
    t5 = diag(tanh(diag(part1)));
    t6 = diag(sech(diag(part1)));
    dt5 = 0.5 * t6 * t6 * Lam2_p12;

    bt = M2' * Lam1_p12 + Lam2_p12 * t5 * M2' * Lam1_n12;
    br = M2' * Lam1_p12 - Lam2_p12 * t5 * M2' * Lam1_n12;

    dbt = Lam2_p12 * dt5 *  M2' *  Lam1_n12;
    dbr = -dbt;

    d_bt1 = - (I/bt) * dbt * (I/bt);
else
    disp('UMRT_Jac_lvl5.m: Wrong choice of Jacobian parameter!!');
end
```

```
%%% Program Name: UMRT_Jac_lvl4
%%% Description: this code is for calculating dref and dtra
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [rf tra d_rf d_tra] =
UMRT_Jac_lvl4(M1,dM1,dM1t,d_zeta1,d_a1,d_bt1,dbr,zeta,a,bt,br,ivar)

tra = 2 * M1 / bt / zeta / a * M1';
rf  = tra - M1 / bt * br * M1';

if (strcmp(ivar,'ks') || strcmp(ivar,'ka') || strcmp(ivar,'Tbot') ||
strcmp(ivar,'slope'))
    tra_p1 = dM1 /  bt /   zeta /  a *   M1';
    tra_p2 =  M1 * d_bt1 / zeta /  a *   M1';
    tra_p3 =  M1 /  bt * d_zeta1 / a *   M1';
    tra_p4 =  M1 /  bt /   zeta * d_a1 * M1';
    tra_p5 =  M1 /  bt /   zeta /  a *  dM1t;

    d_tra = 2 * (tra_p1 + tra_p2 + tra_p3 + tra_p4 + tra_p5);

    rf_p1 = dM1 /  bt   *  br *  M1';
    rf_p2 =  M1 * d_bt1 *  br *  M1';
    rf_p3 =  M1 /  bt   * dbr *  M1';
    rf_p4 =  M1 /  bt   *  br * dM1t;

    d_rf = d_tra - (rf_p1 + rf_p2 + rf_p3 + rf_p4);

elseif (strcmp(ivar,'thickness'))
    tra_p1 =  M1 * d_bt1 / zeta /  a *   M1';
    tra_p2 =  M1 /  bt * d_zeta1 / a *   M1';
    tra_p3 =  M1 /  bt /   zeta * d_a1 * M1';

    d_tra = 2 * (tra_p1 + tra_p2 + tra_p3);

    rf_p1 =  M1 * d_bt1 *  br *  M1';
    rf_p2 =  M1 /  bt   * dbr *  M1';

    d_rf = d_tra - (rf_p1 + rf_p2);
else
    disp('UMRT_Jac_lvl5.m: Wrong choice of Jacobian parameter!!');
end
```

```
%%% Program Name: UMRT_Jac_lvl3_v2
%%% Description:
%%% 1. This code is for computing up- and down- welling radiation vectors and
%%%    their derivatives for a single layer
%%% 2. INPUT:  a. dA, dB, dref, dtra are all [2M, 2M] matrices w.r.t ONE
%%%               parameter
%%%    OUTPUT: a.  ups,  dns: [2M, 1] vectors
%%%            b. dups, ddns: [2M, 1] vectors
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [ups dns dups ddns] =
UMRT_Jac_lvl3_v2(A,B,ref,tra,dA,dB,dref,dtra,weights1,mu,ka,Tbot,Ttop,thickne
ss,radtype,opt,jnum)
h = thickness;
slope = (Ttop - Tbot)/h;
M = length(mu);
I = eye(2*M);
e = I - ref - tra;
d_e = -(dref + dtra);

%%% differentiation of A^-1 and B^-1
Ai = I/A;
Bi = I/B;
d_Ai = - Ai*dA*Ai;
d_Bi = - Bi*dB*Bi;

if (strcmp(radtype,'constant'))
    flayer = sqrt(weights1./mu)*ka*Tbot;
    Flayer = [flayer flayer]';
    upinh_top = Ai*Flayer;        % DOTLRT inh. sol., eq. 40
    ups = e*upinh_top;            % DOTLRT self rad., eq. 41
    dns = ups;

    if jnum == 1                  % jnum = 1, differentiate e, A, w.r.t ks
        dups = d_e*upinh_top + e*d_Ai*Flayer;
        ddns = dups;
    elseif jnum == 2              % jnum = 2, differentiate e, A, F, w.r.t ka
        dflayer = sqrt(weights1./mu)*Tbot;   % ka term is not here
        dFlayer = [dflayer dflayer]';
        p1 = d_e*upinh_top;
        p2 = e*(d_Ai*Flayer + Ai*dFlayer);
        dups = p1 + p2;
        ddns = dups;
    elseif jnum == 3              % jnum = 3, differentiate F w.r.t Tbot
        dflayer = sqrt(weights1./mu)*ka;     % Tbot term is not here
        dFlayer = [dflayer dflayer]';
        dups = e*Ai*dFlayer;
        ddns = dups;
    elseif jnum == 4              % jnum = 4, constant temp.
        dups = zeros(2*M,1);
        ddns = zeros(2*M,1);
    elseif jnum == 5              % jnum = 5, only related to ref and tra
        dups = d_e*upinh_top;
        ddns = dups;
    else
        disp('UMRT_Jac_lvl3.m: Derivatives of this parameter is not included
yet!!! ')
```

```
    end

elseif (strcmp(radtype,'linear'))
    flayer_top = sqrt(weights1./mu)*ka*(Tbot+slope*h);
    flayer_bot = sqrt(weights1./mu)*ka*Tbot;
    fslope = sqrt(weights1./mu)*ka*slope;
    Flayer_top = [flayer_top'; flayer_top'];
    Flayer_bot = [flayer_bot'; flayer_bot'];
    Fslope = [fslope'; fslope'];

    if (opt == 1)
        upinh_top = Ai*Flayer_top - Bi*Ai*Fslope;
        dninh_top = Ai*Flayer_top + Bi*Ai*Fslope;
        upinh_bot = Ai*Flayer_bot - Bi*Ai*Fslope;
        dninh_bot = Ai*Flayer_bot + Bi*Ai*Fslope;
        ups = upinh_top - ref*dninh_top - tra*upinh_bot;
        dns = dninh_bot - ref*upinh_bot - tra*dninh_top;

        if jnum == 1
            up_p1  = -dref*Ai*Flayer_top + (I-ref)*d_Ai*Flayer_top;
            up_p2  = (dtra*Ai + tra*d_Ai)*Flayer_bot;
            up_p3a = (dref - dtra)*Bi*Ai*Fslope;
            up_p3b = (I + ref - tra)*(d_Bi*Ai + Bi*d_Ai)*Fslope;
            up_p3  = up_p3a + up_p3b;
            dn_p1  = -dref*Ai*Flayer_bot + (I-ref)*d_Ai*Flayer_bot;
            dn_p2  = (dtra*Ai + tra*d_Ai)*Flayer_top;
            dn_p3  = up_p3;

            dups = up_p1 - up_p2 - up_p3;
            ddns = dn_p1 - dn_p2 + dn_p3;

        elseif jnum == 2
            dflayer_top = sqrt(weights1./mu)*(Tbot+slope*h);
            dflayer_bot = sqrt(weights1./mu)*Tbot;
            dfslope     = sqrt(weights1./mu)*slope;
            dFlayer_top = [dflayer_top'; dflayer_top'];
            dFlayer_bot = [dflayer_bot'; dflayer_bot'];
            dFslope     = [dfslope'; dfslope'];

            up_p1a = d_Ai*Flayer_top + Ai*dFlayer_top;
            up_p1  = -dref*Ai*Flayer_top + (I-ref)*up_p1a;
            up_p2a = (dtra*Ai + tra*d_Ai) * Flayer_bot;
            up_p2b = tra*Ai*dFlayer_bot;
            up_p2  = up_p2a + up_p2b;
            up_p3a = (dref - dtra)*Bi*Ai*Fslope;
            up_p3b = (I + ref - tra)*(d_Bi*Ai + Bi*d_Ai)*Fslope;
            up_p3c = (I + ref - tra)*(I/B)*(I/A)*dFslope;
            up_p3  = up_p3a + up_p3b + up_p3c;
            dn_p1a = d_Ai*Flayer_bot + Ai*dFlayer_bot;
            dn_p1  = -dref*Ai*Flayer_bot + (I-ref)*dn_p1a;
            dn_p2a = (dtra*Ai + tra*d_Ai)*Flayer_top;
            dn_p2b = tra*Ai*dFlayer_top;
            dn_p2  = dn_p2a + dn_p2b;
            dn_p3  = up_p3;

            dups = up_p1 - up_p2 - up_p3;
```

```
            ddns = dn_p1 - dn_p2 + dn_p3;

        elseif jnum == 3
            dflayer_top = sqrt(weights1./mu)*ka;
            dflayer_bot = sqrt(weights1./mu)*ka;
            dFlayer_top = [dflayer_top'; dflayer_top'];
            dFlayer_bot = [dflayer_bot'; dflayer_bot'];

            dups = e*Ai*dFlayer_top;
            ddns = e*Ai*dFlayer_bot;

        elseif jnum == 4
            dflayer_top = sqrt(weights1./mu)*ka*h;
            dfslope     = sqrt(weights1./mu)*ka;
            dFlayer_top = [dflayer_top'; dflayer_top'];
            dFslope     = [dfslope'; dfslope'];

            up_p1 = (I - ref)*Ai*dFlayer_top;
            up_p2 = zeros(2*M,1);
            up_p3 = (I + ref - tra)*Bi*Ai*dFslope;
            dn_p1 = zeros(2*M,1);
            dn_p2 = tra*Ai*dFlayer_top;
            dn_p3 = up_p3;

            dups = up_p1 - up_p2 - up_p3;
            ddns = dn_p1 - dn_p2 + dn_p3;

        elseif jnum == 5
            dflayer_top = sqrt(weights1./mu)*ka*slope;
            dFlayer_top = [dflayer_top'; dflayer_top'];

            up_p1 = -dref*Ai*Flayer_top + (I - ref)*Ai*dFlayer_top;
            up_p2 = dtra*Ai*Flayer_bot;
            up_p3 = (dref - dtra)*Bi*Ai*Fslope;
            dn_p1 = -dref*Ai*Flayer_bot;
            dn_p2 = dtra*Ai*Flayer_top + tra*Ai*dFlayer_top;
            dn_p3 = up_p3;

            dups = up_p1 - up_p2 - up_p3;
            ddns = dn_p1 - dn_p2 + dn_p3;

        else
            disp('UMRT_Jac_lvl3.m: Derivatives of this parameter is not
included yet!!! ')
        end
    else
        disp('UMRT_Jac_lvl3.m: wrong solution choice.');
    end
else
    disp('UMRT_Jac_lvl3.m: wrong radiation type.');
end
```

```
%%% Program Name: UMRT_ML_RTU3ih
%%% Description:
%%% 1. This program is for calculating the reflection and transmission
%%%    matrices of a Multilayer with/without Refractive interfaces
%%% 2. In UMRT, we categorize:
%%% 2a) Layers:
%%%      Inhomogeneous: internal reflection and transmission: r and t
%%%      homogeneous:   does not have internal reflection and transmission
%%% 2b) Interfaces: inside a multilayer, they are either refractive or not
%%% 2c) Boundary Conditions: Inh.- Homo. or Homo.- Inh. or Inh. - Inh.
%%%
%%% 3. Code Name:
%%% 3a) UMRT_ML_RTUx stands for UMRT, Multilayer, Reflection, Transmission,
%%%     and Upwelling Radiation
%%% 3b) RTU1:    for non-refractive inhomo. layers (same as in DOTLRT)
%%% 3c) RTU2:    for refractive inhomo. layers
%%% 3d) RTU3ii:  up- and dn- R and T for refractive inhomo.- inhomo. layers
%%% 3e) RTU3iim: up- and dn- R and T for refractive inhomo.- inhomo. layers
%%% 3f) RTU3hi:  up- and dn- R and T for refractive   homo.- inhomo. layers
%%% 3g) RTU3ih:  up- and dn- R and T for refractive inhomo.-   homo. layers
%%% 3h) RTU4:    total R, T, U for a stack
%%%
%%% 4. Note:
%%% 4a) all UMRT_ML_RTUs are validated using Energy Conservation and also
%%%     validated by comparing results with another program written under
%%%     DOTLRT/UMRT procedure (see code: TestCode2.m).
%%% 4b) the earlier version UMRT_ML_RTU3ii is no longer in use and
%%%     replaced by UMRT_ML_RTU3iim.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [R01_tet T01_tet R10_tet T10_tet] =
UMRT_ML_RTU3ih(angle,r1,eps0,eps1)
M = length(angle); I = eye(2*M);

%%% Below is modified to fit UMRT
[Tv01 Th01 Rv01 Rh01] = frp3(angle,eps0,eps1);
[Tv10 Th10 Rv10 Rh10] = frp3(angle,eps1,eps0);
T01 = [Tv01 Th01]; T10 = [Tv10 Th10];
R01 = [Rv01 Rh01]; R10 = [Rv10 Rh10];

%%% Cubic Spline Interpolation
[T01_intp] = Interp5(angle,eps0,eps1,T01);
[T10_intp] = Interp5(angle,eps1,eps0,T10);
T01 = diag(T01_intp); T10 = diag(T10_intp);
R01 = diag(R01); R10 = diag(R10);

%%% Matches eqns (8.25 - 8.28) in pp.350, A.F.Kung's book
%%% Note: 1) the book has typos, beware.
%%%       2) the indexes of transmission matrices are different than that
%%%          in A.F.Kung's book, since we interpolate them before use.
R10_tet = R10 / (I - r1*R10);
T10_tet = T10 / (I - r1*R10);

R01_tet = R01 + T10 * r1 * (I / (I - R10*r1)) * T01;
T01_tet =                  (I / (I - R10*r1)) * T01;
```

```
%%% Program Name: UMRT_ML_RTU3hi
%%% Description:
%%% 1. This program is for calculating the reflection and transmission
%%%    matrices of a Multilayer with/without Refractive interfaces
%%% 2. Detailed description can be found in UMRT_ML_RTU3ih
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [R12_tet T12_tet R21_tet T21_tet] =
UMRT_ML_RTU3hi(angle,r1,eps1,eps2)
M = length(angle); I = eye(2*M);

[Tv12 Th12 Rv12 Rh12] = frp3(angle,eps1,eps2);
[Tv21 Th21 Rv21 Rh21] = frp3(angle,eps2,eps1);
T12 = [Tv12 Th12]; T21 = [Tv21 Th21];
R12 = [Rv12 Rh12]; R21 = [Rv21 Rh21];

%%% Cubic Spline Interpolation
[T12_intp] = Interp5(angle,eps1,eps2,T12);
[T21_intp] = Interp5(angle,eps2,eps1,T21);
T12 = diag(T12_intp); T21 = diag(T21_intp);
R12 = diag(R12); R21 = diag(R21);

%%% Matches eqns (8.17 - 8.20) in pp.348, A.F.Kung's book
%%% Note: 1) the book has typos, beware.
%%%       2) the indexes of transmission matrices are different than that
%%%          in A.F.Kung's book, since we interpolate them before use.
R21_tet = R21 + T12 * r1 * (I / (I - R12*r1)) * T21;
T21_tet =                  (I / (I - R12*r1)) * T21;

R12_tet = R12 / (I - r1*R12);
T12_tet = T12 / (I - r1*R12);
```

```
%%% Program Name: UMRT_ML_RTU3iim
%%% Description:
%%% 1. This program is for calculating the reflection and transmission
%%%    matrices of a Multilayer with/without Refractive interfaces
%%% 2. Detailed description can be found in UMRT_ML_RTU3ih
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [R12_hat T12_hat R21_hat T21_hat] =
UMRT_ML_RTU3iim(angle,r1,r2,eps1,eps2)
M = length(angle); I = eye(2*M);

%%% Fresnel's Refractivity and Transmissivity
[Tv12 Th12 Rv12 Rh12] = frp3(angle,eps1,eps2);
[Tv21 Th21 Rv21 Rh21] = frp3(angle,eps2,eps1);
T12 = [Tv12 Th12]; T21 = [Tv21 Th21];
R12 = [Rv12 Rh12]; R21 = [Rv21 Rh21];

%%% Cubic Spline Interpolation
[T12_intp] = Interp5(angle,eps1,eps2,T12);
[T21_intp] = Interp5(angle,eps2,eps1,T21);
T12 = diag(T12_intp); R12 = diag(R12);
T21 = diag(T21_intp); R21 = diag(R21);

Rip21 = (I / (I - R21*r2)) * R21;
Rip12 = (I / (I - R12*r1)) * R12;

Tis21 = T21*r2 / (I - R21*r2) * R21;
Tis12 = T12*r1 / (I - R12*r1) * R12;

Ris21 = T12*r1 / (I - R12*r1) * T21;
Ris12 = T21*r2 / (I - R21*r2) * T12;

Tip21 = (I / (I - R12*r1)) * T21;
Tip12 = (I / (I - R21*r2)) * T12;

Rup21 = I / (I - R21*r2);
Rup12 = I / (I - R12*r1);

Tds21 = T21*r2 / (I - R21*r2);
Tds12 = T12*r1 / (I - R12*r1);

Rus21 = T12*r1 / (I - R12*r1);
Rus12 = T21*r2 / (I - R21*r2);

Tdp21 = I / (I - R12*r1);
Tdp12 = I / (I - R21*r2);

R21_hat = Rip21 + Rup21 / (I - Rus21*Tds21) * Ris21 + Rup21 / (I -
          Rus21*Tds21) * Rus21 * Tis21;
T21_hat = Tip21 + Tdp21 / (I - Tds21*Rus21) * Tis21 + Tdp21 / (I -
          Tds21*Rus21) * Tds21 * Ris21;
R12_hat = Rip12 + Rup12 / (I - Rus12*Tds12) * Ris12 + Rup12 / (I -
          Rus12*Tds12) * Rus12 * Tis12;
T12_hat = Tip12 + Tdp12 / (I - Tds12*Rus12) * Tis12 + Tdp12 / (I -
          Tds12*Rus12) * Tds12 * Ris12;
```

```
%%% Program Name: UMRT_ML_RTU4
%%% Description:
%%% 1. This program is for calculating the reflection and transmission
%%%    matrices of a Multilayer with/without Refractive interfaces
%%% 2. Detailed description can be found in UMRT_ML_RTU3ih
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [R1_dn,T1_dn,U1,Ll1,Lu1,Ld1,U1a,U1b,U1c] =
UMRT_ML_RTU4(t1,u1,v1,Us,T12_hat,T21_hat,R12_hat,R21_hat,R10_tet,T10_tet)

M = length(u1); I = eye(M);

%%% Matches eqns (8.65 - 8.67) in pp.367, A.F.Kung's book
%%% Note: 1) the book has typos, beware.
%%%       2) the indexes of transmission matrices are different than that
%%%          in A.F.Kung's book, since we interpolate them before use.

Lu1 = T12_hat/(I-t1*R10_tet*t1*R12_hat);
Ld1 = Lu1*t1*R10_tet;
Ll1 = Lu1*t1;

%%% R1_dn: eqn.(8.50), T1_dn: eqn.(8.46)
R1_dn = R21_hat + Lu1*t1*R10_tet*t1*T21_hat;
T1_dn = T10_tet/(I-t1*R12_hat*t1*R10_tet)*t1*T21_hat;

U1a = Lu1*u1; U1b = Ld1*v1; U1c = Ll1*Us;
U1 = U1a + U1b + U1c;
```

```
%%% Program Name: UMRT_ML_RTU3iim_Diff1
%%% Description: This program is for calculating derivative matrices of the
%%%              reflection and transmission in a multilayer stack.
%%% Note: This code accommodates to UMRT_ML_RTU3iim, and _Diff1 means in
%%%       two-layer stack, the perturbed layer is the underneath one.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function  [dR12_hat,dT12_hat,dR21_hat,dT21_hat,R12_hat,T12_hat,R21_hat,T21_hat]
= UMRT_ML_RTU3iim_Diff1(angle,r1,r2,dr1,eps1,eps2)

M = length(angle); I = eye(2*M);
%%% Fresnel's Refractivity and Transmissivity
[Tv12 Th12 Rv12 Rh12] = frp3(angle,eps1,eps2);
[Tv21 Th21 Rv21 Rh21] = frp3(angle,eps2,eps1);
T12 = [Tv12 Th12]; T21 = [Tv21 Th21];
R12 = [Rv12 Rh12]; R21 = [Rv21 Rh21];

%%% Cubic Spline Interpolation
[T12_intp] = Interp5(angle,eps1,eps2,T12);
[T21_intp] = Interp5(angle,eps2,eps1,T21);
T12 = diag(T12_intp); R12 = diag(R12);
T21 = diag(T21_intp); R21 = diag(R21);

%%% Calculate R21_hat and T21_hat, based on A.K.Fung, Chapter 8
Rup21 = (I/(I-R21*r2));
Rip21 = (I/(I-R21*r2))*R21;
Tds21 = T21*r2/(I-R21*r2);
Tis21 = T21*r2/(I-R21*r2)*R21;

Tdp21 = (I/(I-R12*r1));
Tip21 = (I/(I-R12*r1))*T21;
Rus21 = T12*r1/(I-R12*r1);
Ris21 = T12*r1/(I-R12*r1)*T21;

R21_hat = Rip21 + Rup21/(I-Rus21*Tds21)*Ris21 + Rup21/(I-
Rus21*Tds21)*Rus21*Tis21;
T21_hat = Tip21 + Tdp21/(I-Tds21*Rus21)*Tis21 + Tdp21/(I-
Tds21*Rus21)*Tds21*Ris21;

%%% Differentiate all parts w.r.t r1
 Tdp21p1 = I-R12* r1;
dTdp21p1 =  -R12*dr1;

dTdp21   = DinvMatrix(Tdp21p1,dTdp21p1);
dTip21   = dTdp21*T21;
dRus21   = T12*(dr1/(I-R12*r1) + r1*dTdp21);
dRis21   = dRus21*T21;

 R21_hat_p1 = I/(I-Rus21*Tds21);
 R21_hatpp1 =    I-Rus21*Tds21;
dR21_hatpp1 =    -dRus21*Tds21;
dR21_hat_p1 = DinvMatrix(R21_hatpp1,dR21_hatpp1);
dR21_hat_s1 = Rup21*(dR21_hat_p1*Ris21 + R21_hat_p1*dRis21);
dR21_hat_s2 = Rup21*(dR21_hat_p1*Rus21 + R21_hat_p1*dRus21)*Tis21;
dR21_hat    = dR21_hat_s1 + dR21_hat_s2;
```

```
 T21_hat_p1 = I/(I-Tds21*Rus21);
 T21_hatpp1 =    I-Tds21*Rus21;
dT21_hatpp1 =    -Tds21*dRus21;
dT21_hat_p1 = DinvMatrix(T21_hatpp1,dT21_hatpp1);
dT21_hat_s1 = (dTdp21*T21_hat_p1 + Tdp21*dT21_hat_p1)*Tis21;
dT21_hat_s2 = (dTdp21*T21_hat_p1 + Tdp21*dT21_hat_p1)*Tds21*Ris21 +
Tdp21*T21_hat_p1*Tds21*dRis21;
dT21_hat    = dTip21 + dT21_hat_s1 + dT21_hat_s2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tdp12 = (I/(I-R21*r2));
Tip12 = (I/(I-R21*r2))*T12;
Rus12 = T21*r2/(I-R21*r2);
Ris12 = T21*r2/(I-R21*r2)*T12;

Rup12 = (I/(I-R12*r1));
Rip12 = (I/(I-R12*r1))*R12;
Tds12 = T12*r1/(I-R12*r1);
Tis12 = T12*r1/(I-R12*r1)*R12;

R12_hat = Rip12 + Rup12/(I-Rus12*Tds12)*Ris12 + Rup12/(I-
Rus12*Tds12)*Rus12*Tis12;
T12_hat = Tip12 + Tdp12/(I-Tds12*Rus12)*Tis12 + Tdp12/(I-
Tds12*Rus12)*Tds12*Ris12;

%%% Differentiate all parts w.r.t r1
dRup12 = dTdp21;
dRip12 = dRup12*R12;                %%% Previous error: mistakely used T12...
dTds12 = dRus21;
dTis12 = dTds12*R12;                %%% Previous error: mistakely used T12...

 R12_hat_p1 = I/(I-Rus12*Tds12);
 R12_hatpp1 =    I-Rus12*Tds12;
dR12_hatpp1 =    -Rus12*dTds12;
dR12_hat_p1 = DinvMatrix(R12_hatpp1,dR12_hatpp1);
dR12_hat_s1 = (dRup12*R12_hat_p1 + Rup12*dR12_hat_p1)*Ris12;
dR12_hat_s2 = (dRup12*R12_hat_p1 + Rup12*dR12_hat_p1)*Rus12*Tis12 +
Rup12*R12_hat_p1*Rus12*dTis12;
dR12_hat    = dRip12 + dR12_hat_s1 + dR12_hat_s2;

 T12_hat_p1 = I/(I-Tds12*Rus12);
 T12_hatpp1 =    I-Tds12*Rus12;
dT12_hatpp1 =    -dTds12*Rus12;
dT12_hat_p1 = DinvMatrix(T12_hatpp1,dT12_hatpp1);
dT12_hat_s1 = Tdp12*(dT12_hat_p1*Tis12 + T12_hat_p1*dTis12);
dT12_hat_s2 = Tdp12*(dT12_hat_p1*Tds12 + T12_hat_p1*dTds12)*Ris12;
dT12_hat    = dT12_hat_s1 + dT12_hat_s2;
```

```
%%% Program Name: UMRT_ML_RTU3iim_Diff2
%%% Description: This program is for calculating derivative matrices of the
%%%              reflection and transmission in a multilayer stack.
%%% Note: This code accommodates to UMRT_ML_RTU3iim, and _Diff2 means in
%%%       two-layer stack, the perturbed layer is the above one.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function  [dR12_hat,dT12_hat,dR21_hat,dT21_hat,R12_hat,T12_hat,R21_hat,T21_hat]
= UMRT_ML_RTU3iim_Diff2(angle,r1,r2,dr2,eps1,eps2)

M = length(angle); I = eye(2*M);
%%% Fresnel's Refractivity and Transmissivity
[Tv12 Th12 Rv12 Rh12] = frp3(angle,eps1,eps2);
[Tv21 Th21 Rv21 Rh21] = frp3(angle,eps2,eps1);
T12 = [Tv12 Th12]; T21 = [Tv21 Th21];
R12 = [Rv12 Rh12]; R21 = [Rv21 Rh21];

%%% Cubic Spline Interpolation
[T12_intp] = Interp5(angle,eps1,eps2,T12);
[T21_intp] = Interp5(angle,eps2,eps1,T21);
T12 = diag(T12_intp); R12 = diag(R12);
T21 = diag(T21_intp); R21 = diag(R21);

%%% Calculate R21_hat and T21_hat, based on A.K.Fung, Chapter 8
Rup21 = (I/(I-R21*r2));
Rip21 = (I/(I-R21*r2))*R21;
Tds21 = T21*r2/(I-R21*r2);
Tis21 = T21*r2/(I-R21*r2)*R21;

Tdp21 = (I/(I-R12*r1));
Tip21 = (I/(I-R12*r1))*T21;
Rus21 = T12*r1/(I-R12*r1);
Ris21 = T12*r1/(I-R12*r1)*T21;

R21_hat = Rip21 + Rup21/(I-Rus21*Tds21)*Ris21 + Rup21/(I-
Rus21*Tds21)*Rus21*Tis21;
T21_hat = Tip21 + Tdp21/(I-Tds21*Rus21)*Tis21 + Tdp21/(I-
Tds21*Rus21)*Tds21*Ris21;

%%% Differentiate all parts w.r.t r2
 Rup21p1 = I-R21* r2;
dRup21p1 =  -R21*dr2;

dRup21   = DinvMatrix(Rup21p1,dRup21p1);
dRip21   = dRup21*R21;
dTds21   = T21*(dr2/(I-R21*r2) + r2*dRup21);
dTis21   = dTds21*R21;

 R21_hat_p1 = I/(I-Rus21*Tds21);
 R21_hatpp1 =    I-Rus21*Tds21;
dR21_hatpp1 =    -Rus21*dTds21;
dR21_hat_p1 = DinvMatrix(R21_hatpp1,dR21_hatpp1);
dR21_hat_s1 = (dRup21*R21_hat_p1 + Rup21*dR21_hat_p1)*Ris21;
dR21_hat_s2 = (dRup21*R21_hat_p1 + Rup21*dR21_hat_p1)*Rus21*Tis21 +
Rup21*R21_hat_p1*Rus21*dTis21;
dR21_hat    = dRip21 + dR21_hat_s1 + dR21_hat_s2;
```

```
 T21_hat_p1 = I/(I-Tds21*Rus21);
 T21_hatpp1 =    I-Tds21*Rus21;
dT21_hatpp1 =    -dTds21*Rus21;
dT21_hat_p1 = DinvMatrix(T21_hatpp1,dT21_hatpp1);
dT21_hat_s1 = Tdp21*(dT21_hat_p1*Tis21 + T21_hat_p1*dTis21);
dT21_hat_s2 = Tdp21*(dT21_hat_p1*Tds21 + T21_hat_p1*dTds21)*Ris21;
dT21_hat    = dT21_hat_s1 + dT21_hat_s2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tdp12 = (I/(I-R21*r2));
Tip12 = (I/(I-R21*r2))*T12;
Rus12 = T21*r2/(I-R21*r2);
Ris12 = T21*r2/(I-R21*r2)*T12;

Rup12 = (I/(I-R12*r1));
Rip12 = (I/(I-R12*r1))*R12;
Tds12 = T12*r1/(I-R12*r1);
Tis12 = T12*r1/(I-R12*r1)*R12;

R12_hat = Rip12 + Rup12/(I-Rus12*Tds12)*Ris12 + Rup12/(I-
Rus12*Tds12)*Rus12*Tis12;
T12_hat = Tip12 + Tdp12/(I-Tds12*Rus12)*Tis12 + Tdp12/(I-
Tds12*Rus12)*Tds12*Ris12;

%%% Differentiate all parts w.r.t r2
dTdp12 = dRup21;
dTip12 = dTdp12*T12;
dRus12 = dTds21;
dRis12 = dRus12*T12;

 R12_hat_p1 = I/(I-Rus12*Tds12);
 R12_hatpp1 =    I-Rus12*Tds12;
dR12_hatpp1 =    -dRus12*Tds12;
dR12_hat_p1 = DinvMatrix(R12_hatpp1,dR12_hatpp1);
dR12_hat_s1 = Rup12*(dR12_hat_p1*Ris12 + R12_hat_p1*dRis12);
dR12_hat_s2 = Rup12*(dR12_hat_p1*Rus12 + R12_hat_p1*dRus12)*Tis12;
dR12_hat    = dR12_hat_s1 + dR12_hat_s2;

 T12_hat_p1 = I/(I-Tds12*Rus12);
 T12_hatpp1 =    I-Tds12*Rus12;
dT12_hatpp1 =    -Tds12*dRus12;
dT12_hat_p1 = DinvMatrix(T12_hatpp1,dT12_hatpp1);
dT12_hat_s1 = (dTdp12*T12_hat_p1 + Tdp12*dT12_hat_p1)*Tis12;
dT12_hat_s2 = (dTdp12*T12_hat_p1 + Tdp12*dT12_hat_p1)*Tds12*Ris12 +
Tdp12*T12_hat_p1*Tds12*dRis12;
dT12_hat    = dTip12 + dT12_hat_s1 + dT12_hat_s2;
```

```
%%% Program Name: UMRT_ML_RTU4_Diff1
%%% Description: This program is for Upwardly calculating total R, T, U and
%%%              their derivatives for a multilayer stack.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [dR1_dn,dT1_dn,dU1,R1_dn,T1_dn,U1,dLl1,Ll1] =
UMRT_ML_RTU4_Diff1(t1,u1,v1,Us,T12_hat,T21_hat,R12_hat,R21_hat,R10_tet,T10_te
t,dUs,dR10_tet,dT10_tet)

M = length(u1); I = eye(M);

Lu1 = T12_hat/(I-t1*R10_tet*t1*R12_hat);
Lu2 = T10_tet/(I-t1*R12_hat*t1*R10_tet);
Ld1 = Lu1*t1*R10_tet;
Ll1 = Lu1*t1;
%%% R1_dn: eqn.(8.50), T1_dn: eqn.(8.46)
R1_dn = R21_hat + Lu1*t1*R10_tet*t1*T21_hat;
T1_dn = Lu2*t1*T21_hat;
U1 = Lu1*u1 + Ld1*v1 + Ll1*Us;

 Lu1p = I-t1* R10_tet*t1*R12_hat;
dLu1p =  -t1*dR10_tet*t1*R12_hat;
 Lu2p = I-t1*R12_hat*t1* R10_tet;
dLu2p =  -t1*R12_hat*t1*dR10_tet;
d_Lu1p = DinvMatrix(Lu1p,dLu1p);
d_Lu2p = DinvMatrix(Lu2p,dLu2p);

dLu1 = T12_hat*d_Lu1p;
dLu2 = dT10_tet/Lu2p + T10_tet*d_Lu2p;
dLd1 = dLu1*t1*R10_tet + Lu1*t1*dR10_tet;
dLl1 = dLu1*t1;

dR1_dn = (dLu1*t1*R10_tet + Lu1*t1*dR10_tet)*t1*T21_hat;
dT1_dn = dLu2*t1*T21_hat;
dU1 = dLu1*u1 + dLd1*v1 + dLl1*Us + Ll1*dUs;
```

```
%%% Program Name: UMRT_ML_RTU4_Diff2
%%% Description: This program is for Downwardly calculating total R, T, U and
%%%              their derivatives for a multilayer stack.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [dR1_dn,dT1_dn,dU1,R1_dn,T1_dn,U1] =
UMRT_ML_RTU4_Diff2(t1,u1,v1,Us,T12_hat,T21_hat,R12_hat,R21_hat,R10_tet,T10_te
t,dR12_hat,dT12_hat,dR21_hat,dT21_hat)

M = length(u1); I = eye(M);

Lu1 = T12_hat/(I-t1*R10_tet*t1*R12_hat);
Lu2 = T10_tet/(I-t1*R12_hat*t1*R10_tet);
Ld1 = Lu1*t1*R10_tet;
Ll1 = Lu1*t1;
%%% R1_dn: eqn.(8.50), T1_dn: eqn.(8.46)
R1_dn = R21_hat + Lu1*t1*R10_tet*t1*T21_hat;
T1_dn = Lu2*t1*T21_hat;
U1 = Lu1*u1 + Ld1*v1 + Ll1*Us;

 Lu1p  = I-t1*R10_tet*t1* R12_hat;
dLu1p  =  -t1*R10_tet*t1*dR12_hat;
 Lu2p  = I-t1* R12_hat*t1*R10_tet;
dLu2p  =  -t1*dR12_hat*t1*R10_tet;
d_Lu1p = DinvMatrix(Lu1p,dLu1p);
d_Lu2p = DinvMatrix(Lu2p,dLu2p);

dLu1 = dT12_hat/Lu1p + T12_hat*d_Lu1p;
dLd1 = dLu1*t1*R10_tet;
dLl1 = dLu1*t1;
dLu2 = T10_tet*d_Lu2p;

dR1_dn = dR21_hat + dLu1*t1*R10_tet*t1*T21_hat + Lu1*t1*R10_tet*t1*dT21_hat;
dT1_dn = dLu2*t1*T21_hat + Lu2*t1*dT21_hat;
dU1 = dLu1*u1 + dLd1*v1 + dLl1*Us;
```

```
%%% Program Name: DinvMatrix
%%% Description: This program is for differentiating an inverse matrix by
%%%              knowing the derivative of that matrix.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function  dA1 = DinvMatrix(A,dA)
I = eye(length(A));
dA1 = -(I/A) * dA * (I/A);
```

```
%%% Program Name: frp3
%%% Description:
%%% 1. This function is used for calculating the bistatic scattering matrix
%%%    for specular surface
%%% 2. Note:
%%% 2a) AJG, ECEN5264 course slide 19: the input of the demonstration
%%%     of (rv,rh) is corrected to be (eps1=1, eps2=4.2) by AJG, 03/17/2011.
%%% 2b) Bistatic definition is correct, validated eqn (22) by setting s = 0.
%%%
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [tv th rv rh] = frp3(angle,eps_low,eps_high)
M = length(angle);

if eps_low == eps_high
    rv = zeros(1,M); rh = zeros(1,M);
    tv = ones(1,M);  th = ones(1,M);
else
    mu = cos(angle); smu = sin(angle); % based on incident angles
    kix = sqrt(eps_low)*smu; kix_re = real(kix);
    kiz = sqrt(eps_low)*mu;  kiz_re = real(kiz);
    ktz = sqrt(eps_high - kix.^2); ktz_re = real(ktz);

    Rh = (kiz - ktz)./(kiz + ktz);
    Rv = (eps_high*kiz - eps_low*ktz)./(eps_high*kiz + eps_low*ktz);

%%% note: the relation among reflectivity, transmissivity, scattering and
%%% emissivity is described in Tsang, vol.I, page 151.
    rv = abs(Rv).^2; rh = abs(Rh).^2;
    tv = ones(1,M) - rv; th = ones(1,M) - rh;
end
```

```
%%% Program Name: Interp5
%%% Description: This program is for interpolation as described in my thesis.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [T_intp Tvf Thf angle_ref] = Interp5(angle,eps_low,eps_high,trans)

M = length(angle);
%%% Find the quadrature angles that can pass through to the next layer
angle2 = angle(find(trans(1:M)>1e-10)); % qudrature angles that can pass
N = length(angle2);

if M == N
    load xk5; load wx5;    % M = 5000
    a1 = 0; b1 = pi/2;     % Note: inc_theta is from 0 to pi/2
    angle1 = a1 + (b1-a1)*(-xk+1)/2;
    weights1 = wx*(b1-a1)/2;

    M_f = length(weights1); % f means finer grids
    [Tvf Thf]=frp3(angle1,eps_low,eps_high);
    mu_inc = cos(angle1);   % based on incident angles
    smu_inc = sin(angle1);

    %%% Snell's Law for general LHI media, given by AJG's course note #19
    kix = sqrt(eps_low)*smu_inc; kix_re = real(kix);
    kiz = sqrt(eps_low)*mu_inc;  kiz_re = real(kiz);
    ktz = sqrt(eps_high - eps_low*smu_inc.^2);
    ktz_re = real(ktz);
    s1 = (kix_re.^2 + kiz_re.^2);
    s2 = (kix_re.^2 + ktz_re.^2);

    angle_ref = real(asin(sin(angle1).*sqrt(s1./s2))); % refraction angles

    %%% Find the quadrature angles inside the refractive-angle-cone
    ind = 1;
    for i = 1:M
        if angle(i) <= max(angle_ref) && angle(i) >= min(angle_ref)
            angle3(ind) = angle(i); % qud-angles inside the angle_ref cone
            ind = ind+1;
        end
    end
    clear ind

    %%% Find the starting # of the refractive quadrature angles
    for i = 1:M
        if angle(i) <= max(angle_ref)
            jj = i;
            break;
        end
    end

    JJ = length(angle3); % determine # of the refractive quadrature angles

    Tvip = spline(angle_ref,Tvf,angle3);
    Thip = spline(angle_ref,Thf,angle3);
    Tv = zeros(1,M); Th = zeros(1,M);
    Tv(jj:jj+JJ-1) = Tvip; Th(jj:jj+JJ-1) = Thip;
```

```
    T_intp = [Tv Th];

else
    theta_ca = fsolve(@(theta_ca) criangle(theta_ca, [eps_high eps_low]), 1e-
9);
    load xk5; load wx5;    % M = 5000
    a1 = 0; b1 = theta_ca; % Note: inc_theta is from 0 to critical angle
    angle1 = a1 + (b1-a1)*(-xk+1)/2;
    weights1 = wx*(b1-a1)/2;

    M_f = length(weights1); % x_f mean x for finer grids
    [Tvf Thf]=frp3(angle1,eps_low,eps_high);

    mu_inc = cos(angle1); smu_inc = sin(angle1);
    %%% Snell's Law for general LHI media, given by AJG's course note #19
    kix = sqrt(eps_low)*smu_inc; kix_re = real(kix);
    kiz = sqrt(eps_low)*mu_inc;  kiz_re = real(kiz);
    ktz = sqrt(eps_high - eps_low*smu_inc.^2);
    ktz_re = real(ktz);
    s1 = (kix_re.^2 + kiz_re.^2);
    s2 = (kix_re.^2 + ktz_re.^2);

    angle_ref = asin(sin(angle1).*sqrt(s1./s2)); % refraction angles

    Tv = spline(angle_ref,Tvf,angle);
    Th = spline(angle_ref,Thf,angle);
    T_intp = [Tv Th];
end
```

```
%%% Program Name: DMRTKs_UMRT
%%% Description:
%%% 1. This program is for calculating DMRT related parameters, which are
%%%    needed in UMRT.
%%% 2. Note:
%%% 2a) the host medium could be water, ice and air.
%%% 2b) the command 'round' is replaced by 'floor'
%%% 2c) pypdffunc is changed, g(r) is from 0 to rm*Dia.
%%% 2d) nk = 1024, rm = 5 is confirmed here.
%%% 2e) (xk1, wx1) are used and same as that in DMRT_PM_GL_Phi
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Ks,Ka,epsilon_p,K_eff,n_max,no,k0fs,k0a,kfs,k] =
DMRTKs_UMRT(freq,Tdiel,Fv,Dia)
tau    = 0.1;               % stickiness
rm     = 5;                 % # of diameter
nk     = 1024;              % # of fourier transform
no     = 6*Fv/(pi*Dia^3);   % particle density

bkgnd  = input('Background medium (occupied more than 50%, chosen by number):
1. Air; 2. Water; 3. Ice. ');
inclu  = input('Scatters material (occupied less than 50%, chosen by number):
1. Air; 2. Water; 3. Ice. ');
material = [bkgnd inclu];

%%%%%%%%%%%%%%%%%% load Gauss-Legender nodes and weights %%%%%%%%%%%%%%%%%%
load xk3; load wx3;         % 16 quadratures
a1 = 0; b1 = pi;            % Theta from 0 to pi, prepared for computing ks.
nodes1   = a1 + (b1-a1)*(-xk+1)/2;
weights1 = wx*(b1-a1)/2;
clear a1 b1 xk wx

%%%%%%%%%%%%%%%%%% compute Percus-Yevick pair distribution %%%%%%%%%%%%%%%%%%
clear pypdf.dat; clear pysf.dat
pypdffunc(Fv,Dia,tau,rm,nk)
load pypdf.dat;
r_b = pypdf(:,1);           % load r_b
gg  = pypdf(:,2);           % load g(r)
r   = r_b*Dia;              % resize r_b w.r.t <D>

%%%%%%%%%%%%%%%%%% load Gauss-Legender nodes and weights %%%%%%%%%%%%%%%%%%
load xk1; load wx1;         % 1000 quadratures
a2 = r(1); b2 = r(end);     % Note: r is from 0 to 5*<D>, prepared for
computing H.
nodes2   = a2 + (b2-a2)*(-xk+1)/2;
weights2 = wx*(b2-a2)/2;
clear a2 b2 xk wx

%%%%%%%%%%%%%%%%%% interpolate PY distribution %%%%%%%%%%%%%%%%%%
ggyy = spline(r,gg,nodes2);

%%%%%%%%%%%%%%%%%%%%%% compute ks, ka of DMRT %%%%%%%%%%%%%%%%%%%%%%
%%% Note: DMRT codes are in unit: Naper/cm:(naper/cm)*100 = naper/m and
%%% (naper/cm)*100*4.343 = dB/m
[Ks,Ka,K_eff,epsilon_p,n_max,k0fs,k0a,kfs,k] =
DMRTKs(freq,Fv,no,Dia,Tdiel,weights1,nodes1,weights2,nodes2,ggyy,material);
```

```
%%% Program Name: DRMTKs
%%% Description:
%%% 1. This program is for calculating DMRT related parameters, which are
%%%    needed in UMRT.
%%% 2. Note:
%%% 2a) the host medium could be water, ice and air.
%%% 2b) the code is based on the general Mie scattering notations from Ulaby
%%%     et al, Vol.I, pp.290
%%% 2c) add a judgment to guarantee epsilon_p is in right formation:
%%%                epsilon_p = epsilon_p' + 1i*epsilon_p"
%%% 2d) qcamie.m is not changed and it is written by Chite Chen.
%%%
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Ks,Ka,K_eff,epsilon_p,n_max,k0fs,k0a,kfs,k] =
DMRTKs(freq,Fv,no,Dia,Tdiel,weights1,nodes1,weights2,nodes2,ggyy,material)
if material(1) == material(2)
    disp('Warning: Scatters should not be the same material as the home
medium!!');
end

if material(1) == 1
    eps_bkgnd = 1;
elseif material(1) == 2
    eps_bkgnd = conj(d3lec(freq,Tdiel,0,2));
elseif material(1) == 3
    eps_bkgnd = conj(h2o_ice_diel(freq,Tdiel));
else
    disp('Background medium is not defined!!');
end

if material(2) == 1
    eps_inclu = 1;
elseif material(2) == 2
    eps_inclu = conj(d3lec(freq,Tdiel,0,2));
elseif material(2) == 3
    eps_inclu = conj(h2o_ice_diel(freq,Tdiel));
else
    disp('Scatter material is not defined!!');
end

%%% General Notations of Mie Scattering From Ulaby et al, Vol.I, pp.290 %%%
epsilon_p = eps_inclu/eps_bkgnd;        % scatter permittivity relative to
                                          homogeneous background
if imag(epsilon_p) >= 0
    epsilon_p = epsilon_p;
else
    epsilon_p = conj(epsilon_p);
end

lambda    = 30/freq;                    % free-space wavelength in cm
k0fs      = pi*Dia/lambda;              % size parameter of free-space
k0a       = k0fs*sqrt(real(eps_bkgnd)); % size parameter of background
                                          (general loss medium)
kfs       = 2*pi/lambda;                % wave number of free-space
k         = kfs*sqrt(real(eps_bkgnd));  % wave number of background
n_max     = floor(k*Dia)+1;             % n_max is for DMRT-QCA computation
```

```
if n_max > 3
    n_max = 3;
else
    n_max = n_max;
end

%%% qcamie.m will load pypdf.dat inside
[K_eff] = qcamie(freq,epsilon_p,Fv,k0fs,k0a,k,n_max);

%%% Note: DMRT_PM.m requires g(r) from 1<D> to infinity.
%%%       AddF.m    requires g(r) from   0  to infinity.
theta  = nodes1;                          % forward scattering angle
for i = 1:length(nodes1)
    Ctheta = theta(i);
    [p11,p22,p33,p34,Ka,Kr] =
DMRT_PM(freq,epsilon_p,Fv,k0fs,k0a,k,n_max,Ctheta,K_eff);
    qq = AddF(ggyy,Kr,k,Ctheta,no,weights2,nodes2);
    P11_num(i) = p11*qq; P22_num(i) = p22*qq;
    qf(i) = qq;
end

Ks = pi*sum(weights1.*(P11_num+P22_num).*sin(theta));
```

```
%%% Program Name: DMRT_PM
%%% Description: This program is for computing the DMRT phase matrix elements
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [p11 p22 p33 p34 Ka Kr] =
DMRT_PM(freq,epsilon_p,Fv,k0fs,k0a,k,n_max,Ctheta,K_eff)
lambda = 30/freq;            % free-space wavelength in cm
kfs    = 2*pi/lambda;        % wave number in free-space
a      = k0fs/kfs ;          % radius in cm: k0fs = pi*2*a/lambda
b      = 2*a;                % diameter in cm
ka     = k0a;                % size parameter of background
kpa    = ka*sqrt(epsilon_p); % kpa is used in the Bessel functions
no     = 6*Fv/(pi*b^3);      % particle density

%%% Read pair function for given Fv for the integration limit of Mp
%%% r_b starts at 0 and extends to 5*<D>.
clear pypdf.dat;
 load pypdf.dat;
r_b = pypdf(:,1);            % r from 0 to 5*<D>
gg  = pypdf(:,2);            % g(r) from PY distribution
r   = r_b*b;                 % resize r
Kr  = real(K_eff);           % Note: this K_eff is from qcamie.m

%%% a) Under K_eff, compute new Matrix of the L-L law by the E-O theorem
T4 = SysEqu(n_max,k,K_eff,ka,kpa,b,no,r,gg);

%%% b) Prepare Tn_M and Tn_N for replacing one row of the above matrix
p1 = -1i*(pi*no)/k^2;
for i = 1:n_max
    tn_m(i) = p1*Tn_M(i,ka,kpa)*(2*i+1);
    tn_n(i) = p1*Tn_N(i,ka,kpa)*(2*i+1);
    TnM(i)  = Tn_M(i,ka,kpa);
    TnN(i)  = Tn_N(i,ka,kpa);
end

%%% c) The NEW matrix of the L-L law by the E-O theorem
row = [tn_m tn_n];
for i = 1:2*n_max
    temp = T4;
    temp(i,:) = row;
    if norm(det(temp)) > 1e-14;
        ind = i;
        flagn = 1;
        break;
    else
        flagn = 0;
    end
end

if flagn == 0
    disp('DMRT-QCA has no solution due to singularity');
else
end

T4_new = temp;
```

```
%%% d) Prepare solution vector
col = zeros(2*n_max, 1);
col(ind) = K_eff - k;

%%% e) Solve for Xn_M and Xn_N
Xn = T4_new \ col;
XnM = Xn(1:n_max);
XnN = Xn(n_max+1:end);

R = 0;
p2 = -p1/(k+Kr);
for i = 1:n_max
    p3 = TnM(i)*XnM(i);
    p4 = TnN(i)*XnN(i);
    R = R + p2*((-1)^(i))*(-p3+p4)*(2*i+1);
end

Ka = 0;
p5 = (k/Kr)*2*pi/(k^2)/((abs(1-R))^2)*no;
for i = 1:n_max
    p6 = (abs(XnM(i)))^2*(-real(TnM(i))-(abs(TnM(i)))^2);
    p7 = (abs(XnN(i)))^2*(-real(TnN(i))-(abs(TnN(i)))^2);
    Ka = Ka + p5*(2*i+1)*(p6+p7);
end

[pin taun]=AngFunc2(Ctheta,n_max);
p8 = -1i/(1-R)*sqrt(1/k/Kr);
f11 = 0;
f22 = 0;
for i = 1:n_max
    p9  = (2*i+1)/(i*(i+1));
    p10 = TnM(i)*XnM(i)*taun(i)+TnN(i)*XnN(i)*pin(i);
    p11 = TnM(i)*XnM(i)*pin(i)+TnN(i)*XnN(i)*taun(i);
    f11 = f11 + p8*p9*p10;
    f22 = f22 + p8*p9*p11;
end

p11 = abs(f11)^2;
p22 = abs(f22)^2;
p33 = real(f11*conj(f22));
p34 = -imag(f11*conj(f22));
end
```

```
%%% Program Name: AngFunc2
%%% Description: This program is for computing the angular dependent function
%%%              pi_n and tau_n used for Mie calculation
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [pin taun] = AngFunc2(theta,n_max)
mu = cos(theta);

%%% Use pi_0 =0 and pi_1 = 1 to first compute pi_2 = 3cos(theta)
%%% Then use pi_1, pi_2, tau_1, tau_2 as starting values for the iteration
pin(1) = 1; pin(2) = 3*mu;
taun(1) = mu; taun(2) = 3*cos(2*theta);

%%% The upward recurrence formulae are given in BH, pp95, eqn.(4.47)
for j = 3:n_max
    pin(j) = (2*j-1)/(j-1)*mu*pin(j-1) - j/(j-1)*pin(j-2);
    taun(j) = j*mu*pin(j) - (j+1)*pin(j-1);
end
end
```

```
%%% Program Name: AddF
%%% Description: This program is for computing the structure factor used for
%%%              DMRT phase matrix calculation
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function qq = AddF(ggyy,Kr,k,Ctheta,no,weights1,nodes1)
pp = sqrt(Kr^2 + k^2 -2*k*Kr*cos(Ctheta));
H  = (1/2/(pi^2)/pp)*sum(weights1.*nodes1.*(ggyy-1).*sin(pp*nodes1));
qq = no*(1 + no*(2*pi)^3 * H);
```

```
%%% Program Name: PolyMieCoeffWaterIce
%%% Description:
%%% 1. This program is for computing the scattering, absorption and
%%%    asymmetric coefficients from polydispersive Mie theory
%%% 2. Note:
%%% 2a) quadrature number should be the same as that in MiePM_R_Phi3.m.
%%% 2b) permittivity should be from the same function (as now, d3lec.m).
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Kg,Kscat,Kabso]=PolyMieCoeffWaterIce(freq,T,mD,PR,SDFopt,material)

load xk3; load wx3;            % 16 Christoffel weights and nodes
a = 0; b = 15*mD;              % Integration limits from 0 to 15*diameter
nodes = a + (b-a)*(-xk+1)/2;
weights = wx*(b-a)/2;

scatnumeval = zeros(1,length(nodes));
absonumeval = zeros(1,length(nodes));
asymnumeval = zeros(1,length(nodes));
Kscat       = zeros(1,length(freq));
Kabso       = zeros(1,length(freq));
Kg          = zeros(1,length(freq));

for i = 1:length(freq)
    f = freq(i);
    for j = 1:length(nodes)
        D = nodes(j);
        [out] = PolyDispNumEval(D,f,T,mD,PR,SDFopt,material);
        scatnumeval(j) = out(1);
        absonumeval(j) = out(2);
        asymnumeval(j) = out(3);
    end

    Kscat(i) = (pi/4)*sum(weights.*scatnumeval)*1e-6; %1e-6: Naper/Meter
    Kabso(i) = (pi/4)*sum(weights.*absonumeval)*1e-6; %1e-6: Naper/Meter
    Kg(i)    = sum(weights.*asymnumeval)./sum(weights.*scatnumeval);
end
```

```
%%% Program Name: PolyDispNumEval
%%% Description:
%%% 1. This program is for computing the scattering, absorption and
%%%    asymmetric coefficients from polydispersive Mie theory
%%% 2. Note: the code is based on
%%% 2a) C.Matzler, 2002 "Matlab Functions for Mie Scattering and Absorption
%%%     Version 2"
%%% 2b) A.Gasiewski, "Microwave Radiative Transfer in Hydrometeors"
%%% 2c) A code written by Sandeep Kumar, CET
%%% 2d) currently, the code includes 9 PDFs.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [out] = PolyDispNumEval(D,f,T,mD,R,SDFopt,material)

%%% This function outputs the integrands for obtaining the scattering,
%%% absorption, and asymmetric coefficients. The integrand outputs are
%%% vectors depending on the input vector 'D' and other scalar parameters.

if SDFopt == 1
    nD = 1000/(pi*(mD^4))*exp(-D/mD);
elseif SDFopt == 2
    Lambda_MP = 4.1*R^(-0.21);      %(mm^-1)
    nD = 8000.*exp(-Lambda_MP.*D);
elseif SDFopt == 3                  %Best rain size distribution
    W = 67*R^(0.846);               %(m^-3)(mm^3)
    a = 1.3*R^(0.232);              %(mm^-1)
    nD = (13.5*W/pi/a^4).*(D./a).^(-1.75).*exp(-(D./a).^(2.25));
elseif SDFopt == 7.1                %JTW rain size distribution
    Lambda_JD = 5.7*R^(-0.21);      %(mm^-1)
    nD = 30000.*exp(-Lambda_JD.*D);
elseif SDFopt == 7.2                %JW, widespread
    Lambda_JW = 4.1*R^(-0.21);      %(mm^-1)
    nD = 7000.*exp(-Lambda_JW.*D);
elseif SDFopt == 7.3                %JT, Thunderstorm
    Lambda_JT = 3.0*R^(-0.21);      %(mm^-1)
    nD = 1400.*exp(-Lambda_JT.*D);
elseif SDFopt == 4                  %AU rain size distribution
    Lambda_AU = 7.09*R^(-0.27);     %(mm^-1)
    nD = 64500*R^(-0.5).*D.^2.*exp(-Lambda_AU.*D);
elseif SDFopt == 5                  %SL rain size distribution
    b = 0.26*R^(0.44);              %mm
    c = 0.95*R^(0.14);
    nD = 1000*(c/b).*(D./b).^(c-1).*exp(-(D./b).^c);
elseif SDFopt == 6
    Lambda_VB = 4.85*R^(-2/9);  %(mm^-1)
    nD = 8000.*exp(-Lambda_VB.*D);
elseif SDFopt == 8
    Lambda_SS = 2.29*R^(-0.45);
    nD = 2500*R^(-0.94)*exp(-Lambda_SS.*D);
elseif SDFopt == 9
    fv = 0.25;
    No = (fv*1e9)/(pi*(mD^4));
    Lambda_dense = 1/mD;
    nD = No*exp(-Lambda_dense.*D);
end

intg1 = nD.* (D.^2);
```

```
%Size parameter x
x = pi*D*f/(3e2); % in mm/GHz
mu = 1;

%%% For Matzlers code e = ep + iepp, so conjugate it taken.
if(strcmp(material,'water'))
%     T = T - 273.15;
%     esp = purewaterpermittiviy(T,f);
%     eps = real(esp)-1i*imag(esp);
      eps = conj(d3lec(f,T,0,2));
elseif (strcmp(material,'ice'))
%     esp = seaicepermittivity(T,f);
%     eps = real(esp)+1i*imag(esp);
      eps = conj(h2o_ice_diel(f,T));
else
    disp('PolyDispMieExpIntegrand.m :: Wrong material. Enter either water or
ice');
    return;
end

result = mie2(eps,mu,x);
Kex    = result(1);
Ksc    = result(2);
g      = result(5);
Kab    = Kex - Ksc;

out1   = intg1 .* Ksc;
out2   = intg1 .* Kab;
out3   = intg1 .* Ksc .* g;
out    = [out1, out2, out3];
end
```

```
%%% Program Name: purewaterpermittiviy
%%% Description:
%%% 1. This program is for computing the permittivity of pure water
%%% 2. The formulae used in this program are from
%%% 2a) "An improved model for the dielectric constant of sea water at
%%%     microwave frequencies", by Klein & Swift, 1976
%%% 2b) "The complex dielectric constant of pure and sea water from microwave
%%%     satellite observations", by Meissner & Matzler, 2004
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function eps_pw = purewaterpermittiviy(T,v)
%%%%%%Double Debye Relation by Wentz and Meissner, 2004%%%%%%%%%%%%%%%%
a0 = 5.7230e0;
a1 = 2.2379e-2;
a2 = -7.1237e-4;
a3 = 5.0478e0;
a4 = -7.0315e-2;
a5 = 6.0059e-4;
a6 = 3.6143e0;
a7 = 2.8841e-2;
a8 = 1.3652e-1;
a9 = 1.4825e-3;
a10 = 2.4166e-4;

%%%Using above parameter to calculate following
eps1 = a0+a1*T+a2*T^2;
v1 = (45+T)/(a3+a4*T+a5*T^2);
epsinf = a6+a7*T;
v2 = (45+T)/(a8+a9*T+a10*T^2);
eps_s = (3.70886e4-8.2168*10*T)/(4.21854*100+T);
eps_pw = zeros(1,length(v));

for i=1:length(v)
    p1 = (eps_s-eps1)/(1+1i*v(i)/v1);
    p2 = (eps1-epsinf)/(1+1i*v(i)/v2);
    eps_pw(i) = p1+p2+epsinf;
end

%%%Stogryn
st_epsinf = 4.9;
st_eps_s = 87.134-1.949e-1*T-1.276e-2*T^2+2.491e-4*T^3;
st_tao = (1.768e-11-6.086e-13*T+1.104e-14*T^2-8.111e-17*T^3)*10^9;

%%%Pure water permittivity
ks_eps_pw = zeros(1,length(v));
gu_eps_pw = zeros(1,length(v));
st_eps_pw = zeros(1,length(v));

for i=1:length(v)
    st_eps_pw(i) = st_epsinf + (st_eps_s-st_epsinf)/(1+2*pi*1i*v(i)*st_tao);
end

%%%Wentz and Meissner parameter
wm_epsinf = 4.44;
wm_eps_s = 87.90*exp(-0.004585*T);
wm_lr = 3.30*exp(-0.0346*T+0.00017*T^2);
```

```
lv = 30./v;
eta = 0.012;
wm_eps_pw = zeros(1,length(v));

for i=1:length(v)
    wm_eps_pw(i) = wm_epsinf + (wm_eps_s-wm_epsinf)/(1+((1i*wm_lr/lv(i))^(1-
eta)));
end
end
```

```
%%% Program Name: seaicepermittivity
%%% Description: The formulae used in this program are from Peter Ray
%%%              "Broadband complex refractive indices of ice and water",
%%%              1972 equations (5), (6), (12)
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function eps_si = seaicepermittivity(t,v)
c = 3e10; %speed of light in cm
lamda = c./v/10^9;

eps_inf = 3.168;
eps_s = 203.168+2.5*t+0.15*t^2;
alpha = 0.288+0.0052*t+0.00023*t^2;
sigma = 1.26*exp(-12500/((t+273.15)*1.9869));
lamda_s = 9.990288e-4*exp(13200/((t+273.15)*1.9869));

p1 = eps_s-eps_inf;
p2 = (lamda_s./lamda).^(1-alpha);
p3 = sin(alpha*pi/2);
p4 = cos(alpha*pi/2);
p5 = (lamda_s./lamda).^(2*(1-alpha));

eps_re = eps_inf+(p1*(1+p2.*p3))./(1+2.*p2.*p3+p5);
eps_im = p1.*p2.*p4./(1+2.*p2.*p3+p5)+sigma.*lamda./(18.8496e10);
eps_si = eps_re+1i*eps_im;
end
```

```
%%% Program Name: KsMieMatrix
%%% Description: This program is for computing the vertical and horizontal
%%% scattering coefficients from integrating the Mie phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Ksv Ksh]=KsMieMatrix(freq,eps,mD,SDFopt,R,scale)
load xk3; load wx3;    % M = 16
a1 = 0; b1 = pi;       % Note: Theta from 0 to pi
nodes1 = a1 + (b1-a1)*(-xk+1)/2;
weights1 = wx*(b1-a1)/2;
smu = sin(nodes1);
M = length(nodes1);
Pvv = zeros(M,M); Pvh = zeros(M,M);
Phv = zeros(M,M); Phh = zeros(M,M);
Ksv = zeros(1,M); Ksh = zeros(1,M);

matlabpool open
for i = 1:M
    theta_s = nodes1(i);    % theta_s is in radian (0 to pi/2)
    parfor j = 1:M
        theta_i = nodes1(j); % theta_i is in radian (0 to pi/2)
        [MiePM]=MiePM_R_Phi3(theta_s,theta_i,freq,eps,mD,SDFopt,R);
        MiePM= MiePM/1e6*scale;
        Pvv(i,j) = MiePM(1,1); Pvh(i,j) = MiePM(2,1);
        Phv(i,j) = MiePM(1,2); Phh(i,j) = MiePM(2,2);
    end
    Ksv(i) = sum((Pvv(i,:)+Pvh(i,:)).*smu.*weights1);
    Ksh(i) = sum((Phv(i,:)+Phh(i,:)).*smu.*weights1);
end
matlabpool close
```

```
%%% Program Name: MiePM_R_Phi3
%%% Description: This program is for computing the reduced Mie phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [MiePM]=MiePM_R_Phi3(theta_s,theta_i,freq,eps,mD,SDFopt,R)

%%% Based on given frequency (in GHz) and permittivity
lambda = 3e2./freq;      % wave length (in cm)
k      = 2*pi/lambda;    % wave number
m      = sqrt(eps);      % used in Mie coefficient calculation

%%% load Gauss-Legender nodes and weights
load xk3; load wx3;
a1 = 0; b1 = 2*pi;                 % Phi from 0 to 2*pi
nodes1   = a1 + (b1-a1)*(-xk+1)/2; % since my xk is from [1 -1], not [-1 1]
weights1 = wx*(b1-a1)/2;
clear xk wx a1 b1

load xk3; load wx3;
a2 = 0; b2 = 15*mD;                % D from 0 to 15*mD
nodes2   = a2 + (b2-a2)*(-xk+1)/2;
weights2 = wx*(b2-a2)/2;
clear xk wx a2 b2

nD   = zeros(1,length(nodes2));
x    = zeros(1,length(nodes2));
nmax = zeros(1,length(nodes2));
p11  = zeros(length(nodes1),length(nodes2));
p22  = zeros(length(nodes1),length(nodes2));
p33  = zeros(length(nodes1),length(nodes2));
p34  = zeros(length(nodes1),length(nodes2));

for i = 1:length(nodes1)
    phi = nodes1(i);
    [Li Ls]=TranMatrix(theta_s, theta_i, phi, 0);  %%% 02/09/2011
    for j = 1:length(nodes2)
        D = nodes2(j);
        if SDFopt == 1
            No = 1000/(pi*(mD^4));
            nD(j) = No*exp(-D/mD);
        elseif SDFopt == 2  %MP rain size distribution
            No = 8000;
            Lambda_MP = 4.1*R^(-0.21);    %(mm^-1)
            nD(j) = No.*exp(-Lambda_MP.*D);
        elseif SDFopt == 3  %Best rain size distribution
            No = 1;
            W = 67*R^(0.846);            %(m^-3)(mm^3)
            a = 1.3*R^(0.232);           %(mm^-1)
            nD(j) = (13.5*W/pi/a^4).*(D./a).^(-1.75).*exp(-(D./a).^(2.25));
        elseif SDFopt == 7.1             %JW rain size distribution
            No = 30000;
            Lambda_JD = 5.7*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JD.*D);
        elseif SDFopt == 7.2             %JW, widespread
            No = 7000;
            Lambda_JW = 4.1*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JW.*D);
```

```
        elseif SDFopt == 7.3            %JW, Thunderstorm
            No = 1400;
            Lambda_JT = 3.0*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JT.*D);
        elseif SDFopt == 4              %AU rain size distribution
            No = 64500*R^(-0.5);
            Lambda_AU = 7.09*R^(-0.27);  %(mm^-1)
            nD(j) = No.*D.^2.*exp(-Lambda_AU.*D);
        elseif SDFopt == 5              %SL rain size distribution
            No = 1000;
            b = 0.26*R^(0.44);          %mm
            c = 0.95*R^(0.14);
            nD(j) = No*(c/b).*(D./b).^(c-1).*exp(-(D./b).^c);
        elseif SDFopt == 6
            No = 8000;
            Lambda_VB = 4.85*R^(-2/9);  %(mm^-1)
            nD(j) = No.*exp(-Lambda_VB.*D);
        elseif SDFopt == 8
            No = 2500*R^(-0.94);
            Lambda_SS = 2.29*R^(-0.45);
            nD(j) = No.*exp(-Lambda_SS.*D);
        elseif SDFopt == 9
            fv = 0.25;
            No = (fv*1e9)/(pi*(mD^4));
            Lambda_dense = 1/mD;
            nD(j) = No*exp(-Lambda_dense.*D);
        end

        x(j) = pi*D/lambda;
        nmax(j) = round(2+x(j)+4*x(j)^(1/3));
        [pin taun] = AngFunc_GL(theta_s,theta_i,phi,nmax(j)); % 02/09/2011
        [Mie_an Mie_bn] = MieSC(m,x(j));

        n = (1:nmax(j));
        n2 = (2*n+1)./(n.*(n+1));
        pn = n2.*pin;
        tn = n2.*taun;

        s_1 = sum(Mie_an.*pn+Mie_bn.*tn);
        s_2 = sum(Mie_an.*tn+Mie_bn.*pn);
        f11 = (1i/k)*s_1;
        f22 = (1i/k)*s_2;
        p11(i,j) = abs(f11)^2;
        p22(i,j) = abs(f22)^2;
        p33(i,j) = real(f11*conj(f22));
        p34(i,j) = -imag(f11*conj(f22));
    end

    mP_11(i) = sum(weights2.*p11(i,:).*nD);
    mP_22(i) = sum(weights2.*p22(i,:).*nD);
    mP_33(i) = sum(weights2.*p33(i,:).*nD);
    mP_34(i) = sum(weights2.*p34(i,:).*nD);
    LPL = Ls*[mP_11(i) 0 0 0;0 mP_22(i) 0 0;0 0 mP_33(i) mP_34(i);0 0 -
mP_34(i) mP_33(i)]*Li;

    mP_11t(i) = LPL(1,1); mP_12t(i) = LPL(1,2); mP_13t(i) = LPL(1,3);
mP_14t(i) = LPL(1,4);
```

```
    mP_21t(i) = LPL(2,1); mP_22t(i) = LPL(2,2); mP_23t(i) = LPL(2,3);
mP_24t(i) = LPL(2,4);
    mP_31t(i) = LPL(3,1); mP_32t(i) = LPL(3,2); mP_33t(i) = LPL(3,3);
mP_34t(i) = LPL(3,4);
    mP_41t(i) = LPL(4,1); mP_42t(i) = LPL(4,2); mP_43t(i) = LPL(4,3);
mP_44t(i) = LPL(4,4);
end

mP11 = sum(weights1.*mP_11t); mP12 = sum(weights1.*mP_12t); mP13 =
sum(weights1.*mP_13t); mP14 = sum(weights1.*mP_14t);
mP21 = sum(weights1.*mP_21t); mP22 = sum(weights1.*mP_22t); mP23 =
sum(weights1.*mP_23t); mP24 = sum(weights1.*mP_24t);
mP31 = sum(weights1.*mP_31t); mP32 = sum(weights1.*mP_32t); mP33 =
sum(weights1.*mP_33t); mP34 = sum(weights1.*mP_34t);
mP41 = sum(weights1.*mP_41t); mP42 = sum(weights1.*mP_42t); mP43 =
sum(weights1.*mP_43t); mP44 = sum(weights1.*mP_44t);

MiePM = [mP11 mP12 mP13 mP14; mP21 mP22 mP23 mP24; mP31 mP32 mP33 mP34; mP41
mP42 mP43 mP44;];
```

```
%%% Program Name: TranMatrix
%%% Description:
%%% 1. This program is for coordinate rotation: from the forward-scattering
%%%    angle based coordinate system to the conventional coordinate system
%%% 2. It is based on the eqns. in "Radiative Transfer" by Chandrasekhar and
%%%    "Wave Propagation and Scattering in Random Media", pp.36 by A.Ishimaru
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Li Ls mu]=TranMatrix(theta_i,theta_s,phi,opt)

%%% The law of cosines: using theta_i, theta_s and phi.
%%% smu = sine(capital theta) is always positive in this definition
mu = cos(theta_i)*cos(theta_s)+sin(theta_i)*sin(theta_s)*cos(phi);
smu = sqrt(1-mu^2);

%%% Cosines and sines of the surface angle i2, which faces theta_i
cos_i2 = (cos(theta_i)*sin(theta_s)-cos(theta_s)*sin(theta_i)*cos(phi))/smu;
if phi > 0 && phi < pi          % eqn (3.24) in pp.113, CM book
    sa2 = acos(cos_i2);
else
    sa2 = 2*pi-acos(cos_i2);
end
sin_i2 = sin(sa2);
cos_2i2 = 2*cos_i2^2-1;
sin_2i2 = 2*sin_i2*cos_i2;

%%% Cosines and sines of the surface angle i1, which faces theta_s
cos_i1 = (cos(theta_s)*sin(theta_i)-cos(theta_i)*sin(theta_s)*cos(phi))/smu;
if phi > 0 && phi < pi
    sa1 = acos(cos_i1);
else
    sa1 = 2*pi-acos(cos_i1);
end
sin_i1 = sin(sa1);
cos_2i1 = 2*cos_i1^2-1;
sin_2i1 = 2*sin_i1*cos_i1;

%%% All elements of the L matrix (scattering angle, i2)
Ls11 = cos_i2^2; Ls12 = sin_i2^2; Ls13 = -sin_2i2/2; Ls14 = 0;
Ls21 = Ls12;     Ls22 = Ls11;     Ls23 = -Ls13;      Ls24 = 0;
Ls31 = sin_2i2;  Ls32 = -Ls31;    Ls33 = cos_2i2;    Ls34 = 0;
Ls41 = 0;        Ls42 = 0;        Ls43 = 0;          Ls44 = 1;

%%% All elements of the L matrix (incident angle, i1)
Li11 = cos_i1^2; Li12 = sin_i1^2; Li13 = -sin_2i1/2; Li14 = 0;
Li21 = Li12;     Li22 = Li11;     Li23 = -Li13;      Li24 = 0;
Li31 = sin_2i1;  Li32 = -Li31;    Li33 = cos_2i1;    Li34 = 0;
Li41 = 0;        Li42 = 0;        Li43 = 0;          Li44 = 1;

if opt == 0
    Li = [Li11 Li12 Li13 Li14;Li21 Li22 Li23 Li24;Li31 Li32 Li33 Li34;Li41
Li42 Li43 Li44;];
    Ls = [Ls11 Ls12 Ls13 Ls14;Ls21 Ls22 Ls23 Ls24;Ls31 Ls32 Ls33 Ls34;Ls41
Ls42 Ls43 Ls44;];
else
end
```

```
end
%%% Program Name: AngFunc_GL
%%% Description: This program is for computing the angular dependent function
%%%              pi_n and tau_n used for Mie calculation in the form of
%%%              vectors that are directly related to Gauss-Legendre
%%%              quadrature angles
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [pin taun mu]=AngFunc_GL(theta,theta_p,phi,nmax)

%%% The law of cosines: cosine and sine of the scattering angle (capital
%%% theta) based on the central angles/sides: theta, theta_p and phi.

mu = cos(theta)*cos(theta_p)+sin(theta)*sin(theta_p)*cos(phi);

%%% Use pi_0 =0 and pi_1 = 1 to first compute pi_2 = 3cos(theta)
%%% Then use pi_1, pi_2, tau_1, tau_2 as starting values for the iteration
pin(1) = 1; pin(2) = 3*mu;
taun(1) = mu; taun(2) = 3*(2*mu^2-1);

%%% The upward recurrence formulae are given in BH, pp95, eqn.(4.47)
for j = 3:nmax
    pin(j) = (2*j-1)/(j-1)*mu*pin(j-1) - j/(j-1)*pin(j-2);
    taun(j) = j*mu*pin(j) - (j+1)*pin(j-1);
end
```

```
%%% Program Name: MieSC
%%% Description: This program is for calculating the Mie Scattering
%%%              Coefficients, an, bn of orders from n=1 to n_max by giving
%%%              the size parameter, x and complex refractive index, m
%%% The code is based on the formulae given by
%%% 2a) C.Matzler, 2002 "Matlab Functions for Mie Scattering and Absorption
%%%     Version 2"
%%% 2b) A.Gasiewski, "Microwave Radiative Transfer in Hydrometeors"
%%% 2c) L.Tsang, "Scattering of Electromagnetic Waves, Vol.I"
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [Mie_an Mie_bn nmax] = MieSC(m,x)
z = m.*x;                              %m*x is used many times in the formulae
nmax = round(2+x+4*x.^(1/3));          %C.Matzler, pp3, eqn.(p.477).
nmx = round(max(nmax,abs(z))+16);      %C.Matzler, pp4.

n = 1:nmax;           %order of the Bessel function.
ns = n+0.5;           %relationship of the order between spherical Bessel and
                      %Bessel functions is: n+0.5.

xs = sqrt(0.5*pi./x); %also from the relationship above
sbess1 = xs.*besselj(ns,x);  %1st kind of spherical Bessel from 1st kind of
                              Bessel, C.Matzler, pp3. (4.9).
sbess2 = xs.*bessely(ns,x);  %2nd kind of spherical Bessel from 2nd kind of
                              Bessel, C.Matzler, pp3. (4.10).
shank1 = sbess1+1i*sbess2;   %1st kind of Hankel function.

p1x = [sin(x)./x, sbess1(1:nmax-1)];   %Phi_(n-1)_(x),from 0 to (n-1)th order
ch1x = [-cos(x)./x, sbess2(1:nmax-1)]; %p1x, ch1x
dnx(nmx)=0+0i;         %C.Matzler, pp4. Dn(z)=0 where n=nmx.
for j = nmx:-1:2       %downward recurrence from n = nmx to 2.
    dnx(j-1) = j./z-1/(dnx(j)+j./z);
end;

dn = dnx(n);           %Dn(z), n= 1 to nmax.
da = dn.*m+n./x;
db = dn./m+n./x;

%%% The definitions of an and bn given by A.Gasiewski are different with
%%% C.Matzler's definitions: an(AJG) = -bn(CM); bn(AJG) = -an(CM)
%%% ---AJG's definitions---
% Mie_an=-(da.*sbess1-p1x)./(da.*shank1-gs1x);
% Mie_bn=-(db.*sbess1-p1x)./(db.*shank1-gs1x);

%%% ---CM's definitions---
Mie_an=(db.*sbess1-p1x)./(db.*shank1-gs1x);
Mie_bn=(da.*sbess1-p1x)./(da.*shank1-gs1x);
end
```

```
%%% Program Name: Mie_NRPM
%%% Description: This program is for computing the Normalized Reduced Mie
%%%              phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [MiePM]=Mie_NRPM(theta_s,theta_i,freq,eps,mD,SDFopt,R,ksv,ksh)

lambda = 3e2./freq;      % wave length (in cm)
k      = 2*pi/lambda;    % wave number
m      = sqrt(eps);      % used in Mie coefficient calculation

%%% load Gauss-Legender nodes and weights
load xk3; load wx3;
a1 = 0; b1 = 2*pi;                 % Phi from 0 to 2*pi
nodes1   = a1 + (b1-a1)*(-xk+1)/2; % since my xk is from [1 -1], not [-1 1]
weights1 = wx*(b1-a1)/2;
clear xk wx a1 b1

load xk3; load wx3;
a2 = 0; b2 = 15*mD;                % D from 0 to 15*mD
nodes2   = a2 + (b2-a2)*(-xk+1)/2;
weights2 = wx*(b2-a2)/2;
clear xk wx a2 b2

nD   = zeros(1,length(nodes2));
x    = zeros(1,length(nodes2));
nmax = zeros(1,length(nodes2));
p11  = zeros(length(nodes1),length(nodes2));
p22  = zeros(length(nodes1),length(nodes2));
p33  = zeros(length(nodes1),length(nodes2));
p34  = zeros(length(nodes1),length(nodes2));

for i = 1:length(nodes1)
    phi = nodes1(i);
    [Li Ls]=TranMatrix(theta_s, theta_i, phi, 0);  %%% 02/09/2011
    for j = 1:length(nodes2)
        D = nodes2(j);
        if SDFopt == 1
            No = 1000/(pi*(mD^4));
            nD(j) = No*exp(-D/mD);
        elseif SDFopt == 2  %MP rain size distribution
            No = 8000;
            Lambda_MP = 4.1*R^(-0.21);    %(mm^-1)
            nD(j) = No.*exp(-Lambda_MP.*D);
        elseif SDFopt == 3  %Best rain size distribution
            No = 1;
            W = 67*R^(0.846);            %(m^-3)(mm^3)
            a = 1.3*R^(0.232);           %(mm^-1)
            nD(j) = (13.5*W/pi/a^4).*(D./a).^(-1.75).*exp(-(D./a).^(2.25));
        elseif SDFopt == 7.1             %JW rain size distribution
            No = 30000;
            Lambda_JD = 5.7*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JD.*D);
        elseif SDFopt == 7.2             %JW, widespread
            No = 7000;
            Lambda_JW = 4.1*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JW.*D);
```

```
        elseif SDFopt == 7.3            %JW, Thunderstorm
            No = 1400;
            Lambda_JT = 3.0*R^(-0.21);   %(mm^-1)
            nD(j) = No.*exp(-Lambda_JT.*D);
        elseif SDFopt == 4              %AU rain size distribution
            No = 64500*R^(-0.5);
            Lambda_AU = 7.09*R^(-0.27);  %(mm^-1)
            nD(j) = No.*D.^2.*exp(-Lambda_AU.*D);
        elseif SDFopt == 5              %SL rain size distribution
            No = 1000;
            b = 0.26*R^(0.44);          %mm
            c = 0.95*R^(0.14);
            nD(j) = No*(c/b).*(D./b).^(c-1).*exp(-(D./b).^c);
        elseif SDFopt == 6
            No = 8000;
            Lambda_VB = 4.85*R^(-2/9);  %(mm^-1)
            nD(j) = No.*exp(-Lambda_VB.*D);
        elseif SDFopt == 8
            No = 2500*R^(-0.94);
            Lambda_SS = 2.29*R^(-0.45);
            nD(j) = No.*exp(-Lambda_SS.*D);
        elseif SDFopt == 9
            fv = 0.25;
            No = (fv*1e9)/(pi*(mD^4));
            Lambda_dense = 1/mD;
            nD(j) = No*exp(-Lambda_dense.*D);
        end

        x(j) = pi*D/lambda;
        nmax(j) = round(2+x(j)+4*x(j)^(1/3));
        [pin taun] = AngFunc_GL(theta_s,theta_i,phi,nmax(j)); % 02/09/2011
        [Mie_an Mie_bn] = MieSC(m,x(j));

        n = (1:nmax(j));
        n2 = (2*n+1)./(n.*(n+1));
        pn = n2.*pin;
        tn = n2.*taun;

        s_1 = sum(Mie_an.*pn+Mie_bn.*tn);
        s_2 = sum(Mie_an.*tn+Mie_bn.*pn);
        f11 = (1i/k)*s_1;
        f22 = (1i/k)*s_2;
        p11(i,j) = abs(f11)^2;
        p22(i,j) = abs(f22)^2;
        p33(i,j) = real(f11*conj(f22));
        p34(i,j) = -imag(f11*conj(f22));
    end

    mP_11(i) = sum(weights2.*p11(i,:).*nD);
    mP_22(i) = sum(weights2.*p22(i,:).*nD);
    mP_33(i) = sum(weights2.*p33(i,:).*nD);
    mP_34(i) = sum(weights2.*p34(i,:).*nD);
    LPL = Ls*[mP_11(i) 0 0 0;0 mP_22(i) 0 0;0 0 mP_33(i) mP_34(i);0 0 -
mP_34(i) mP_33(i)]*Li;

    mP_11t(i) = LPL(1,1); mP_12t(i) = LPL(1,2); mP_13t(i) = LPL(1,3);
mP_14t(i) = LPL(1,4);
```

```
    mP_21t(i) = LPL(2,1); mP_22t(i) = LPL(2,2); mP_23t(i) = LPL(2,3);
mP_24t(i) = LPL(2,4);
    mP_31t(i) = LPL(3,1); mP_32t(i) = LPL(3,2); mP_33t(i) = LPL(3,3);
mP_34t(i) = LPL(3,4);
    mP_41t(i) = LPL(4,1); mP_42t(i) = LPL(4,2); mP_43t(i) = LPL(4,3);
mP_44t(i) = LPL(4,4);
end

mP11 = sum(weights1.*mP_11t); mP12 = sum(weights1.*mP_12t); mP13 =
sum(weights1.*mP_13t); mP14 = sum(weights1.*mP_14t);
mP21 = sum(weights1.*mP_21t); mP22 = sum(weights1.*mP_22t); mP23 =
sum(weights1.*mP_23t); mP24 = sum(weights1.*mP_24t);
mP31 = sum(weights1.*mP_31t); mP32 = sum(weights1.*mP_32t); mP33 =
sum(weights1.*mP_33t); mP34 = sum(weights1.*mP_34t);
mP41 = sum(weights1.*mP_41t); mP42 = sum(weights1.*mP_42t); mP43 =
sum(weights1.*mP_43t); mP44 = sum(weights1.*mP_44t);

MiePM = [mP11/ksv, mP12/ksh; mP21/ksv, mP22/ksh];
```

```
%%% Program Name: HGPM
%%% Description: This program is for computing the Henyey-Greenstein (HG)
%%%              phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [HG]=HGPM(theta_s,theta_i,g)
load xk3; load wx3;
a1 = 0; b1 = pi;                      %AJG, (3.40) is 0 to pi...
nodes1 = a1 + (b1-a1)*(-xk+1)/2;
weights1 = wx*(b1-a1)/2;

cosm = cos(nodes1);
p1 = (1-g^2)/2/pi;
p2 = 1+g^2;
p3 = -2*g*cos(theta_i)*cos(theta_s);
p4 =  2*g*sin(theta_i)*sin(theta_s);

p5 = 1./((p2+p3+p4.*cosm).^(3/2));
s1 = p1*sum(weights1.*p5);
HG = [s1 0;0 s1;];
```

```
%%% Program Name: Rayleigh
%%% Description: This program is for computing the Rayleigh phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [RH]=Rayleigh(theta_s,theta_i)
p1 = sin(theta_i)*sin(theta_s);
p2 = cos(theta_i)*cos(theta_s);
p3 = (3/8)*(1+0.5*p1^2+p2^2);

RH = [p3 0;0 p3;];
```

```
%%% Program Name: DMRT_NRPM
%%% Description: This program is for computing the Normalized Reduced DMRT
%%%              phase matrix
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [DMRTnrpm,Ka] =
DMRT_NRPM(freq,epsilon_p,Fv,k0fs,k0a,kfs,k,n_max,no,theta_s,theta_i,Dia,Ks)
DMRTrpm = zeros(4,4);

load pypdf.dat
r_b = pypdf(:,1); gg = pypdf(:,2); r = r_b*Dia;

%%% load Gauss-Legender nodes and weights (1000)
load xk1; load wx1;
a1 = r(1); b1 = r(end);
nodes1   = a1 + (b1-a1)*(-xk+1)/2;
weights1 = wx*(b1-a1)/2;
clear a1 b1 wx xk

ggyy = spline(r,gg,nodes1);

%%% load Gauss-Legender nodes and weights (16)
%%% Note: Phi from 0 to 2*pi
load xk3; load wx3;
a2 = 0; b2 = 2*pi;
nodes2   = a2 + (b2-a2)*(-xk+1)/2;
weights2 = wx*(b2-a2)/2;
clear a2 b2 wx xk

for i = 1:length(nodes2)
    phi = nodes2(i);
    [Li Ls]=TranMatrix(theta_s,theta_i,phi,0);
    [pin taun mu]=AngFunc_GL(theta_s,theta_i,phi,n_max);
    [p11 p22 p33 p34 Ka Kr] =
DMRT_PM_GL(freq,epsilon_p,Fv,k0fs,k0a,kfs,k,n_max,pin,taun);
    qq = AddF_GL(ggyy,Kr,k,mu,no,weights1,nodes1);
    P11 = p11*qq; P22 = p22*qq;
    P33 = p33*qq; P34 = p34*qq;
    DMRT = [P11 0 0 0; 0 P22 0 0; 0 0 P33 P34; 0 0 -P34 P33];
    DMRTr = Ls*DMRT*Li;
    P11t(i) =  DMRTr(1,1); P12t(i) =  DMRTr(1,2); P13t(i) =  DMRTr(1,3);
P14t(i) =  DMRTr(1,4);
    P21t(i) =  DMRTr(2,1); P22t(i) =  DMRTr(2,2); P23t(i) =  DMRTr(2,3);
P24t(i) =  DMRTr(2,4);
    P31t(i) =  DMRTr(3,1); P32t(i) =  DMRTr(3,2); P33t(i) =  DMRTr(3,3);
P34t(i) =  DMRTr(4,4);
    P41t(i) =  DMRTr(4,1); P42t(i) =  DMRTr(4,2); P43t(i) =  DMRTr(4,3);
P44t(i) =  DMRTr(4,4);
end

DMRTrpm(1,1) = sum(weights2.*P11t); DMRTrpm(1,2) = sum(weights2.*P12t);
DMRTrpm(1,3) = sum(weights2.*P13t); DMRTrpm(1,4) = sum(weights2.*P14t);
DMRTrpm(2,1) = sum(weights2.*P21t); DMRTrpm(2,2) = sum(weights2.*P22t);
DMRTrpm(2,3) = sum(weights2.*P23t); DMRTrpm(2,4) = sum(weights2.*P24t);
DMRTrpm(3,1) = sum(weights2.*P31t); DMRTrpm(3,2) = sum(weights2.*P32t);
DMRTrpm(3,3) = sum(weights2.*P33t); DMRTrpm(3,4) = sum(weights2.*P34t);
```

```
DMRTrpm(4,1) = sum(weights2.*P41t); DMRTrpm(4,2) = sum(weights2.*P42t);
DMRTrpm(4,3) = sum(weights2.*P43t); DMRTrpm(4,4) = sum(weights2.*P44t);

DMRTnrpm = [DMRTrpm(1,1)/Ks, DMRTrpm(1,2)/Ks; DMRTrpm(2,1)/Ks,
DMRTrpm(2,2)/Ks];
```

```
%%% Program Name: DMRT_PM_GL
%%% Description: This program is for computing the DMRT phase matrix, which
%%%              is directly related to the Gauss-Legendre quadrature angles
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function  [p11 p22 p33 p34 Ka Kr] =
DMRT_PM_GL(freq,epsilon_p,Fv,k0fs,k0a,kfs,k,n_max,pin,taun)
a      = k0fs/kfs ;          % radius in cm: k0fs = pi*2*a/lambda
b      = 2*a;                % diameter in cm
ka     = k0a;                % size parameter of background
kpa    = ka*sqrt(epsilon_p); % kpa is used in the Bessel functions
no     = 6*Fv/(pi*b^3);      % particle density

%%% Read pair function for given Fv for the integration limit of Mp
load pypdf.dat;
r_b = pypdf(:,1);
gg  = pypdf(:,2);
r   = r_b*b;                        % resize r;

[K_eff] = qcamie(freq,epsilon_p,Fv,k0fs,k0a,k,n_max);
Kr      = real(K_eff);

%%% a) Under K_eff, compute new Matrix of the L-L law by the E-O theorem
T4 = SysEqu(n_max,k,K_eff,ka,kpa,b,no,r,gg);

%%% b) Form coefficients in eqn.(3)
p1 = -1i*(pi*no)/k^2;
for i = 1:n_max
    tn_m(i) = p1*Tn_M(i,ka,kpa)*(2*i+1);
    tn_n(i) = p1*Tn_N(i,ka,kpa)*(2*i+1);
    TnM(i)  = Tn_M(i,ka,kpa);
    TnN(i)  = Tn_N(i,ka,kpa);
end

%%% c) The NEW matrix of the L-L law by the E-O theorem
row = [tn_m tn_n];
for i = 1:2*n_max
    temp = T4;
    temp(i,:) = row;
    if norm(det(temp)) > 1e-14;
        ind = i;
        flagn = 1;
        break;
    else
        flagn = 0;
    end
end

if flagn == 0
    disp('DMRT-QCA has no solution due to singularity');
else
end

T4_new = temp;

%%% d) Prepare solution vector
```

```
col = zeros(2*n_max, 1);
col(ind) = K_eff - k;

%%% e) Solve for Xn_M and Xn_N
Xn = T4_new \ col;
XnM = Xn(1:n_max);
XnN = Xn(n_max+1:end);

R = 0;
p2 = -p1/(k+Kr);
for i = 1:n_max
    p3 = TnM(i)*XnM(i);
    p4 = TnN(i)*XnN(i);
    R = R + p2*((-1)^(i))*(-p3+p4)*(2*i+1);
end

Ka = 0;
p5 = (k/Kr)*2*pi/(k^2)/(abs(1-R))^2*no;
for i = 1:n_max
    p6 = (abs(XnM(i)))^2*(-real(TnM(i))-(abs(TnM(i)))^2);
    p7 = (abs(XnN(i)))^2*(-real(TnN(i))-(abs(TnN(i)))^2);
    Ka = Ka + p5*(2*i+1)*(p6+p7);
end

p8 = -1i/(1-R)*sqrt(1/k/Kr);
f11 = 0;
f22 = 0;
for i = 1:n_max
    p9  = (2*i+1)/(i*(i+1));
    p10 = TnM(i)*XnM(i)*taun(i)+TnN(i)*XnN(i)*pin(i);
    p11 = TnM(i)*XnM(i)*pin(i)+TnN(i)*XnN(i)*taun(i);
    f11 = f11 + p8*p9*p10;
    f22 = f22 + p8*p9*p11;
end

p11 = abs(f11)^2;
p22 = abs(f22)^2;
p33 = real(f11*conj(f22));
p34 = -imag(f11*conj(f22));
end
```

```
%%% Program Name: weights_yak
%%% Description:
%%% 1. This program is for calculating the Gauss-Legendre weights by giving
%%%    non-negative integer N
%%% 2. It is based on "Accurate Computation of Weights in Classical Gauss-
%%%    Christoffel Quadrature Rules", 1996 by E. Yakimiw
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [wx, xk] = weights_yak(n)
xk = glpt2(n);
px_xk = zeros(1,n+1);
px = zeros(1,n);
dpx = zeros(1,n);
wx = zeros(1,n);

for i = 1:length(xk)
    px_xk(1) = 1;       %P_0(xk(i)) = 1; No matter what x is.
    px_xk(2) = xk(i);   %P_1(xk(i)) = xk(i);
    for j = 2:n         %starting the three terms recurrence, (22) pp.413
    p1 = (2*(j-1)+1)*xk(i)*px_xk(j);
    p2 = (j-1)*px_xk(j-1);
    px_xk(j+1) = (p1-p2)/j;
    end

    px(i) = px_xk(end);                             %P_n(x(i))
    dpx(i) = n*(px_xk(n)-xk(i)*px(i))/(1-xk(i)^2); %(P_n(x(i)))', (22) pp.413

    c2 = 1-xk(i)^2;   %c^2 = 1-x^2
    nb = n*(n+1);

    sigma0 = 1;         % (34-35) pp.415-416
    sigma1 = -xk(i);
    sigma2 = nb*c2+1;
    sigma3 = xk(i)*(nb*c2-1);
    sigma4 = 1-2*nb*c2-nb*c2^2*(3*nb+2);
    sigma5 = -xk(i)*(1-6*nb*c2+nb*c2^2*(17*nb+6));

    f0 = px(i)/dpx(i);
    f0c = f0/c2;
    d0 = sigma0;
    d1 = sigma1*f0c;
    d2 = sigma2/2*(f0c^2);
    d3 = sigma3/6*(f0c^3);
    d4 = sigma4/24*(f0c^4);
    d5 = sigma5/120*(f0c^5);
    wx(i) = 2/c2/((dpx(i)*(d0+d1+d2+d3+d4+d5))^2);
end
```

```
%%% Program Name: glpt2
%%% Description:
%%% 1. This program is for calculating the Gauss-Legendre points (nodes) by
%%%    giving non-negative integer N. It euqals to:
%%%                 true(glpt) = intial(glpt) + correction
%%% 2. It is based on "Accurate Computation of Weights in Classical Gauss-
%%%    Christoffel Quadrature Rules", 1996 by E. Yakimiw
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function xk = glpt2(n)
xk_ini = glpt(n);            %INITIAL values of xk
px_xk = zeros(1,n+1);
px = zeros(1,n);
dpx = zeros(1,n);
Dx = zeros(1,n);

for i = 1:length(xk_ini)
    px_xk(1) = 1;            %P_0(xk(i)) = 1; No matter what x is.
    px_xk(2) = xk_ini(i);    %P_1(xk(i)) = xk(i);
    for j = 2:n              %starting the 3 terms recurrence, (22) pp.413
    p1 = (2*(j-1)+1)*xk_ini(i)*px_xk(j);
    p2 = (j-1)*px_xk(j-1);
    px_xk(j+1) = (p1-p2)/j;
    end

    px(i) = px_xk(end);                                          %P_n(x(i))
    dpx(i) = n*(px_xk(n)-xk_ini(i)*px(i))/(1-xk_ini(i)^2); %(22) pp.413
    f0 = px(i)/dpx(i);                                           %f0, (21) pp. 413

    c2 = 1-xk_ini(i)^2;     %c^2 = 1-x^2
    nb = n*(n+1);
    f0c = f0/c2;

    sig1 = 1;               % (26) pp.414
    sig2 = 2*xk_ini(i);
    sig3 = 2*(2-(3+nb)*c2);
    sig4 = 4*xk_ini(i)*(2-(6+5*nb)*c2);
    sig5 = 4*(4-2*(15+16*nb)*c2+(30+43*nb+6*nb^2)*c2^2);

    Dx(i) = -
c2*((sig1*f0c)+(sig2/2*f0c^2)+(sig3/6*f0c^3)+(sig4/24*f0c^4)+(sig5/120*f0c^5)
); %eqn.(24). pp.414
end
xk = xk_ini + Dx;
```

```
%%% Program Name: pypdffunc
%%% Description:
%%% 1. This program is for calculating the Percus Yevick pair distribution
%%% function and structure factor for a medium with spherical particles
%%% 2. The main code is written by K.H. Ding, 11/1998.
%%% 3. Comment added by Miao Tian for his own research clearance.
%%% 4. In this code, g(r) is produced from (0 to inf) and (1<D> to inf)
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function pypdffunc(fv, dia, tau, rm, nk)
dk   = pi/(rm*dia);
dr   = rm*dia/(nk+1);
hk   = zeros(2*nk+2,1);
vol  = pi*dia^3/6;             %"sphere volume" by (15), Baxter, 1968
rho  = fv/vol;                 %"dimensionless density" by (16), Baxter, 1968
xi   = fv;
xi1  = 1-xi;
nu   = (tau+xi/xi1);              %Quadratic formula: -b = tau+fv/(1-fv)
gama = xi*(1+xi/2)/(3*xi1^2);     %Quadratic formula: 4ac
lamd = 6*(nu-sqrt(nu^2-gama))/xi; %lamd = (-b-sqrt(b^2-4ac))/(2a)
mu   = lamd*xi*xi1;               %mu in (8.4.18) is used in (8.4.24) to
if mu > 1+2*xi                    %guarantee t is the right solution
    return;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Solve Eqn (8.4.19) for the case p=0                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% cst1,2,3 are the coefficients used in (8.4.19) by Tsang, Vol.2, pp.426
%%% Here they are evaluated at pk=0. note: pk is p is Tsang's notation and
%%% when pk=0, psix=sin(x)/x=1, phix=3*(sin(x)/x^3-cos(x)/x^2)=3*(1/3)=1,
%%% and x*phix=0
cst1 = xi/xi1;                     %cst1: fv/(1-fv)
cst2 = 1-lamd*xi+3*cst1;           %cst2: 1-t*fv+3*fv/(1-fv)
cst3 = 3-lamd*xi1;                 %cst3: 3-t*(1-fv)
fpk  = fopen('pysf.dat','w+');
pk   = 0;
pp1  = cst1*(4-lamd+3*cst1)+1;     %1st part of R.S. of (8.4.19)
pp2  = 0;                          %2nd part is 0 when pk=0
pyhk = (1/(pp1^2+pp2^2)-1)/rho;    %H_tet(P)
pysf = 1+pyhk*rho;                 %structure factor
fprintf(fpk,'%6u %14.9f \n',pk,pysf);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Solve Eqn (8.4.19) for the case p~=0                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for ik = 1:nk
  pk   = ik*dk;
  x    = pk*dia/2;
  snx  = sin(x);
  csx  = cos(x);
  psix = snx/x;
  phix = 3*(snx-x*csx)/(x^3);
  pp1  = cst1*(cst2*phix+cst3*psix)+csx;
  pp2  = cst1*x*phix+snx;
  pyhk = (1/(pp1^2+pp2^2)-1)/rho;
  pysf = 1+pyhk*rho;
```

```
  fprintf(fpk,'%6u %14.9f \n',pk,pysf);
  hk(ik+1) = pk*pyhk;
end
fclose(fpk);

hw = -2*fft(hk);
hr = imag(hw(2:nk+1));

fpr = fopen('pypdf.dat','w+');
for ir = 1:nk
  r = ir*dr/dia;
  if r < 1
      g = 0;
      fprintf(fpr,'%14.9f %14.9f \n',r,g);
  elseif r >= 1
      g = 1+hr(ir)/(rm*4*pi*r*dia^2);
      fprintf(fpr,'%14.9f %14.9f \n',r,g);
  end
end
fclose(fpr);
```

```
%%% Program Name: qcamie
%%% Description:
%%% 1. This program is for computing the effective propagation constant using
%%%    the quasi-crystalline approximation (QCA) for a medium consisting of
%%%    densely distributed Mie scatterers
%%% 2. INPUT:
%%% 2a) freq      = frequency in GHz
%%% 2b) epsilon_p = particle permittivity relative to homogeneous background
%%% 2c) f         = fractional volume of particles
%%% 2d) k0a       = size parameter (this can be a vector)
%%% 2e) n_max     = maximum spherical multipole used
%%%
%%%     OUTPUT:
%%% 2f) K_eff     = complex number (per cm) which denote the effective
%%%                 propagation constant at each k0a
%%% 3. This program was original written by Chite Chen, November 1998 and it
%%%    is modified by Miao Tian for his own research needs.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function K_eff=qcamie(freq,epsilon_p,f,k0fs,k0a,k,n_max)
tol    = 1e-14;               % error tolerance of det(T)
lambda = 30/freq;             % free-space wavelength in cm
kfs    = 2*pi/lambda;         % wave number of free-space,
na     = max(size(k0a));

%%% Read pair function for given f for the integration limit of Mp
%%% Mp is integrated from r at 1 diameter to infinity (>= 5 diameter)
clear pypdf.dat; load pypdf.dat;
r_b = pypdf(:,1);
gg  = pypdf(:,2);

for ia = 1:na,
  ka  = k0a(ia);              % size parameter of background: k0a =
k0fs*sqrt(real(eps_bkgnd));
  a   = k0fs /kfs ;           % particle radius in cm
  b   = 2*a;                  % particle diameter in cm
  kpa = ka*sqrt(epsilon_p);   % kpa is used in the Bessel functions
  no  = 6*f/(pi*b^3);         % particle density

%%% First initial guess is the solution for media with sparse concentration
%%% the exciting field approximately the same as the incident field.
%%% K_F means K under Foldy's approximation.
  FF = 0;
  for nn = 1:n_max
    FF = FF+(2*nn+1)*(Tn_M(nn,ka,kpa)+Tn_N(nn,ka,kpa));
  end
  K_F = k-1i*pi*no/k^2*FF;    % (6.1.62) in vol.3, pp.261

%%% Second initial guess is the low frequency limit solution
%%% For Percus-Yevick pair function, (6.1.61) in vol.3, pp.259
%%% a is the radius of the particle of permittivity eps_s.
  y     = (epsilon_p-1)/(epsilon_p+2); % (6.1.52) in vol.3, pp.258
  K_low = sqrt(k^2 + 3*f*k^2*y/(1-f*y) * (1 + 1i*2/3*(k*a)^3*y*(1-f)^4/((1-
f*y)*(1+2*f)^2)));

%%% Third initial guess: K_low_real=real(K_low);
```

```
  r  = r_b*b;                                  % resize r;
  x1 = K_F;                                    % fisrt solution
  x2 = real(K_low);                            % second solution
  x3 = K_low;                                  % third solution
  T1 = SysEqu(n_max,k,x1,ka,kpa,a,no,r,gg);
  T2 = SysEqu(n_max,k,x2,ka,kpa,a,no,r,gg);
  T3 = SysEqu(n_max,k,x3,ka,kpa,a,no,r,gg);
  f1 = det(T1);
  f2 = det(T2);
  f3 = det(T3);

%%% Rearrange the order so that abs(f(i))<= abs(f(i-1)) <= abs(f(i-2))
  [x1 x2 x3 f1 f2 f3]=Rearrange(x1,x2,x3,f1,f2,f3);
  if abs(f3)<tol,
    K_eff(ia)=x3;
  else
%%% Use Muller's Approach to calculate the new guesses
    n_iter=0;                          % number of iteration
    while (abs(f3)>tol)&&(n_iter<=30),
      x_new=Muller(x1,x2,x3,f1,f2,f3);
      x1=x2;
      x2=x3;
      x3=x_new;
      T1=T2;
      T2=T3;
      T3=SysEqu(n_max,k,x3,ka,kpa,a,no,r,gg);
      f1=f2;
      f2=f3;
      f3=det(T3);
      [x1 x2 x3 f1 f2 f3]=Rearrange(x1,x2,x3,f1,f2,f3);
      n_iter=n_iter+1;
    end
    K_eff(ia)=x3;
  end
  K_eff;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  Tn_M.m                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Tn_M(n,ka,kpa)
% compute T-matrix elements for vector spherical waves M_mn

numerator1=sbesselj(n,kpa).*(sbesselj(n,ka)+ka*sbesselj_p(n,ka));
numerator2=sbesselj(n,ka).*(sbesselj(n,kpa)+kpa*sbesselj_p(n,kpa));
denominator1=sbesselj(n,kpa).*(sbesselh(n,ka)+ka*sbesselh_p(n,ka));
denominator2=sbesselh(n,ka).*(sbesselj(n,kpa)+kpa*sbesselj_p(n,kpa));
output=-(numerator1-numerator2)./(denominator1-denominator2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  Tn_N.m                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Tn_N(n,ka,kpa)
% compute T-matrix elements for vector spherical waves N_mn

numerator1=(kpa)^2*sbesselj(n,kpa).*(sbesselj(n,ka)+ka*sbesselj_p(n,ka));
numerator2=(ka)^2*sbesselj(n,ka).*(sbesselj(n,kpa)+kpa*sbesselj_p(n,kpa));
```

```
denominator1=(kpa)^2*sbesselj(n,kpa).*(sbesselh(n,ka)+ka*sbesselh_p(n,ka));
denominator2=(ka)^2*sbesselh(n,ka).*(sbesselj(n,kpa)+kpa*sbesselj_p(n,kpa));
output=-(numerator1-numerator2)./(denominator1-denominator2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  Clebsch.m                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Clebsch(j1,j2,j3,m1,m2,m)
% calculate Clebsch-Gordan coefficients using the formula in
% Abramowitz and Stegun

FF=0;
if (j1 < abs(m1)) || (j2 < abs(m2)) || (j3 < abs(m)),
  output=0;
% sprintf('Condition (j1>=|m1|, j2 >=|m2| and j >=|m|) does NOT obey')
  return
elseif ((j3 > j1+j2) || j3 < abs(j1-j2)),
  output=0;
% sprintf('Condition ( |j1-j2|<=j<=j1+j2 ) does NOT obey')
  return
end

if (m1+m2)~= m,
  output=0;
else
  term1=1/2*(faclog(j1+j2-j3)+faclog(j3+j1-j2)+faclog(j3+j2-j1) ...
       +log(2*j3+1)-faclog(j3+j1+j2+1)+faclog(j1+m1)+faclog(j1-m1) ...
       +faclog(j2+m2)+faclog(j2-m2)+faclog(j3+m)+faclog(j3-m));
% term1 includes the terms which are not related to k
% determine the range for k - the factorial cannot be negative
  upperlimit=min([j1+j2-j3 j1-m1 j2+m2]);
  lowerlimit=abs(min([j3-j2+m1 j3-j1-m2 0]));
  for k=lowerlimit:upperlimit,
    term2=-(faclog(k)+faclog(j1+j2-j3-k)+faclog(j1-m1-k) ...
         +faclog(j2+m2-k)+faclog(j3-j2+m1+k)+faclog(j3-j1-m2+k));
    FF=FF+(-1)^k*exp(term2);
  end
  output=FF*exp(term1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  wigner.m                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=wigner(j1,j2,j3,m1,m2,m)
% calculate Wigner 3-j symbol

output=(-1)^(j1-j2-m)*(2*j3+1)^(-1/2)*Clebsch(j1,j2,j3,m1,m2,-m);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 factorial.m                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=factorial(n)
% compute the factorial of n

product=1;
if n==0,
```

```
  output=1;
elseif (n < 0),
  sprintf('n cannot be negative')
  return
else
  for n_index=1:n,
    product=n_index*product;
  end
  output=product;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  faclog.m                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=faclog(n)
% natural logarithm of factorial n (log(n!)) preventing overflow

nn=0;
if (n==0),
  output=0;                          % log(1) = 0;
elseif (n<0),
  sprintf('n cannot be negative')
  return
else
  for n_index=1:n,
     nn=nn+log(n_index);
  end
  output=nn;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 sbesselj.m                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=sbesselj(n,arg)
% spherical Bessel function of order n
% allow arguement to be an array, but only allow a single order n

output=sqrt(pi./(2*arg)).*besselj(n+1/2,arg);

% special handle for arguement = 0
[xx]=find (arg==0);
if (n==0),
  output([xx])=1;
else
  output([xx])=0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                sbesselj_p.m                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=sbesselj_p(n,arg)
% derivative of spherical Bessel function of order n

output=1/(2*n+1)*(n*sbesselj(n-1,arg)-(n+1)*sbesselj(n+1,arg));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                              sbesselh.m                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=sbesselh(n,arg)
% spherical Hankel function of the first kind of order n
% allow arguement to be an array
% singular at arguement = 0

output = sqrt(pi./(2*arg)).*besselh(n+1/2,1,arg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             sbesselh_p.m                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=sbesselh_p(n,arg)
% derivative of spherical Hankel function of order n

output=1/(2*n+1)*(n*sbesselh(n-1,arg)-(n+1)*sbesselh(n+1,arg));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               a_mnuvp.m                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=a_mnuvp(m,n,u,v,p)
% coefficient a(

output=(-1)^(m+u)*(2*p+1)*sqrt((factorial(n+m)*factorial(v+u)*factorial(p-m-
u)) ...
      /(factorial(n-m)*factorial(v-u)*factorial(p+m+u)))*wigner(n,v,p,m,u,-
(m+u)) ...
      *wigner(n,v,p,0,0,0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              a_mnuvpq.m                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=a_mnuvpq(m,n,u,v,p,q)
% coefficient a(

output=(-1)^(m+u)*(2*p+1)*sqrt((factorial(n+m)*factorial(v+u)*factorial(p-m-
u)) ...
      /(factorial(n-m)*factorial(v-u)*factorial(p+m+u)))*wigner(n,v,p,m,u,-
(m+u)) ...
      *wigner(n,v,q,0,0,0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                A_nvp.m                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=A_nvp(n,v,p)
% coefficient A(

output=1/(n*(n+1)*(2*v+1))*(2*v*(v+1)*(2*v+1)+(v+1)*(n+v-p)*(n+p-v+1) ...
      -v*(n+v+p+2)*(v+p-n+1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                B_nvp.m                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=B_nvp(n,v,p)
% coefficient B(
```

```
output=1/(n*(n+1))*sqrt((n+v+p+1)*(v+p-n)*(n+p-v)*(n+v-p+1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  Lp.m                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Lp(p,k,Keff,b)
% coefficient L

output=-b^2/(Keff^2-k^2)*(k*sbesselh_p(p,k*b)*sbesselj(p,Keff*b) ...
      -Keff*sbesselh(p,k*b)*sbesselj_p(p,Keff*b));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  Mp2.m                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Mp2(p,k,Keff,r,gg,b)
% Pair function is independent of p
% exclude those for r less than b
% find the cutoff point

nr=max(size(r));
r_cutoff=min(find(r>=b));
r=r(r_cutoff:nr);
gg=gg(r_cutoff:nr);
h=gg-1;
hp=sbesselh(p,k*r);
jp=sbesselj(p,Keff*r);
yy=r.^2.*h.*hp.*jp;
output=trapz(r,yy);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 Muller.m                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=Muller(x1,x2,x3,f1,f2,f3)
% Muller's approach

lambda_i=(x3-x2)/(x2-x1);
delta_i=1+lambda_i;
c_i=f1*lambda_i^2-f2*delta_i^2+f3*(lambda_i+delta_i);
den1=c_i+sqrt(c_i^2-4*f3*delta_i*lambda_i*(f1*lambda_i-f2*delta_i+f3));
den2=c_i-sqrt(c_i^2-4*f3*delta_i*lambda_i*(f1*lambda_i-f2*delta_i+f3));
if abs(den1)>=abs(den2),
  denominator=den1;
else
  denominator=den2;
end
output=x3+(x3-x2)*(-2*f3*delta_i)/denominator;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 SysEqu.m                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [output]=SysEqu(n_max,k,Keff,ka,kpa,b,no,r,gg)
% calculate Mp+Lp and store as an array.
% abs(n-v) <=p <=(n+v)
% the upper bound and lower bound are 0 and 2*n_max
```

```
for v=1:n_max,
  for n=1:n_max,
    FF1=0;
    FF2=0;
    for p=abs(n-v):(n+v);
      MLSUM=Lp(p,k,Keff,b)+Mp2(p,k,Keff,r,gg,b);
      aA=a_mnuvp(1,n,-1,v,p)*A_nvp(n,v,p);
      aB=a_mnuvpq(1,n,-1,v,p,p-1)*B_nvp(n,v,p);
      FF1=FF1+MLSUM*aA;
      FF2=FF2+MLSUM*aB;
    end
    MM(v,n)=-2*pi*no*(2*n+1)*Tn_M(n,ka,kpa)*FF1;
    MN(v,n)=-2*pi*no*(2*n+1)*Tn_N(n,ka,kpa)*FF2;
    NM(v,n)=-2*pi*no*(2*n+1)*Tn_M(n,ka,kpa)*FF2;
    NN(v,n)=-2*pi*no*(2*n+1)*Tn_N(n,ka,kpa)*FF1;
  end
end
TT=[MM MN; NM NN];
output=eye(2*n_max)-TT;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Rearrange.m                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x1,x2,x3,f1,f2,f3]=Rearrange(x1,x2,x3,f1,f2,f3)
% random input order, rearrange the order so that f1 >= f2 >= f3

if (abs(f1)<abs(f2)),
  tmp=f2;
  tmpX=x2;
  f2=f1;
  x2=x1;
  f1=tmp;
  x1=tmpX;              % if (f1<f2) interchange f1 and f2 so f1 >= f2
end

if (abs(f2)<abs(f3)),
  tmp=f3;
  tmpX=x3;
  f3=f2;
  x3=x2;
  f2=tmp;
  x2=tmpX;
end

if (abs(f1)<abs(f2)),
  tmp=f2;
  tmpX=x2;
  f2=f1;
  x2=x1;
  f1=tmp;
  x1=tmpX;
end
output=[x1 x2 x3 f1 f2 f3];
```

```
%%% Program Name: h2o_ice_diel
%%% Description:
%%% 1. This program is for computing the permittivity of ice (water)
%%% 2. It is written by Sandeep Kumar, CET.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.
function [kappa] = h2o_ice_diel(freq, temp)

format long e;

c = 2.99793e8;
temp_data = [-1,-5,-20,-60];

wl = [1.670E-1,1.778E-1,1.884E-1,1.995E-1,2.113E-1,2.239E-1,2.371E-1,2.512E-
1,2.661E-1,...
      2.818E-1,2.985E-1, 3.162E-1, 3.548E-1, 3.981E-1, 4.467E-1, 5.012E-
1,5.623E-1, 6.310E-1,...
      7.943E-1, 1.000E0 , 1.259E0, 2.500E0, 5.000E0 , 1.000E1 , 2.000E1 ,
3.200E1,...
      3.500E1,  4.000E1 , 4.500E1 , 5.000E1 , 6.000E1 , 7.000E1 , 9.000E1 ,
1.110E2,...
      1.200E2,  1.300E2 , 1.400E2, 1.500E2, 1.600E2, 1.700E2, 1.800E2,
2.000E2, 2.500E2,...
      2.900E2, 3.200E2];

re = [1.8296, 1.8296, 1.8296, 1.8296 ; 1.8326, 1.8326, 1.8326, 1.8326;...
      1.8315, 1.8315, 1.8315, 1.8315 ; 1.8275, 1.8275, 1.8275, 1.8275;...
      1.8222, 1.8222, 1.8222, 1.8222 ; 1.8172, 1.8172, 1.8172, 1.8172;...
      1.8120, 1.8120, 1.8120, 1.8120 ; 1.8070, 1.8070, 1.8070, 1.8070;...
      1.8025, 1.8025, 1.8025, 1.8025 ; 1.7983, 1.7983, 1.7983, 1.7983;...
      1.7948, 1.7948, 1.7948, 1.7948 ; 1.7921, 1.7921, 1.7921, 1.7921;...
      1.7884, 1.7884, 1.7884, 1.7884 ; 1.7860, 1.7860, 1.7860, 1.7860;...
      1.7843, 1.7843, 1.7843, 1.7843 ; 1.7832, 1.7832, 1.7832, 1.7832;...
      1.7825, 1.7825, 1.7825, 1.7825 ; 1.7820, 1.7820, 1.7820, 1.7820;...
      1.7817, 1.7817, 1.7816, 1.7815 ; 1.7816, 1.7816, 1.7814, 1.7807;...
      1.7819, 1.7819, 1.7816, 1.7801 ; 1.7830, 1.7830, 1.7822, 1.7789;...
      1.7843, 1.7843, 1.7831, 1.7779 ; 1.7852, 1.7852, 1.7838, 1.7773;...
      1.7862, 1.7861, 1.7839, 1.7772 ; 1.7866, 1.7863, 1.7840, 1.7772;...
      1.7868, 1.7864, 1.7840, 1.7772 ; 1.7869, 1.7865, 1.7840, 1.7772;...
      1.7870, 1.7865, 1.7840, 1.7772 ; 1.7870, 1.7865, 1.7840, 1.7772;...
      1.7871, 1.7865, 1.7839, 1.7772 ; 1.7871, 1.7865, 1.7838, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ; 1.7872, 1.7865, 1.7837, 1.7772;...
      1.7872, 1.7865, 1.7837, 1.7772 ];

im = [8.300E-2, 8.300E-2, 8.300E-2, 8.300E-2 ; 6.900E-2, 6.900E-2, 6.900E-2,
6.900E-2;...
      5.700E-2, 5.700E-2, 5.600E-2, 5.700E-2 ; 4.560E-2, 4.560E-2, 4.560E-2,
4.450E-2;...
      3.790E-2, 3.790E-2, 3.790E-2, 3.550E-2 ; 3.140E-2, 3.140E-2, 3.140E-2,
2.910E-2;...
      2.620E-2, 2.620E-2, 2.620E-2, 2.440E-2 ; 2.240E-2, 2.240E-2, 2.190E-2,
1.970E-2;...
      1.960E-2, 1.960E-2, 1.880E-2, 1.670E-2 ; 1.760E-2, 1.760E-2, 1.660E-2,
```

```
1.400E-2;...
      1.665E-2, 1.665E-2, 1.540E-2, 1.235E-2 ; 1.620E-2, 1.600E-2, 1.470E-2,
1.080E-2;...
      1.550E-2, 1.500E-2, 1.350E-2, 8.900E-3 ; 1.470E-2, 1.400E-2, 1.250E-2,
7.340E-3;...
      1.390E-2, 1.310E-2, 1.150E-2, 6.400E-3 ; 1.320E-2, 1.230E-2, 1.060E-2,
5.600E-3;...
      1.250E-2, 1.150E-2, 9.770E-3, 5.000E-3 ; 1.180E-2, 1.080E-2, 9.010E-3,
4.520E-3;...
      1.060E-2, 9.460E-3, 7.660E-3, 3.680E-3 ; 9.540E-3, 8.290E-3, 6.520E-3,
2.990E-3;...
      8.560E-3, 7.270E-3, 5.540E-3, 2.490E-3 ; 6.210E-3, 4.910E-3, 3.420E-3,
1.550E-3;...
      4.490E-3, 3.300E-3, 2.100E-3, 9.610E-4 ; 3.240E-3, 2.220E-3, 1.290E-3,
5.950E-4;...
      2.340E-3, 1.490E-3, 7.930E-4, 3.690E-4 ; 1.880E-3, 1.140E-3, 5.700E-4,
2.670E-4;...
      1.740E-3, 1.060E-3, 5.350E-4, 2.510E-4 ; 1.500E-3, 9.480E-4, 4.820E-4,
2.290E-4;...
      1.320E-3, 8.500E-4, 4.380E-4, 2.110E-4 ; 1.160E-3, 7.660E-4, 4.080E-4,
1.960E-4;...
      8.800E-4, 6.300E-4, 3.500E-4, 1.730E-4 ; 6.950E-4, 5.200E-4, 3.200E-4,
1.550E-4;...
      4.640E-4, 3.840E-4, 2.550E-4, 1.310E-4 ; 3.400E-4, 2.960E-4, 2.120E-4,
1.130E-4;...
      3.110E-4, 2.700E-4, 2.000E-4, 1.060E-4 ; 2.940E-4, 2.520E-4, 1.860E-4,
9.900E-5;...
      2.790E-4, 2.440E-4, 1.750E-4, 9.300E-5 ; 2.700E-4, 2.360E-4, 1.660E-4,
8.730E-5;...
      2.640E-4, 2.300E-4, 1.560E-4, 8.300E-5 ; 2.580E-4, 2.280E-4, 1.490E-4,
7.870E-5;...
      2.520E-4, 2.250E-4, 1.440E-4, 7.500E-5 ; 2.490E-4, 2.200E-4, 1.350E-4,
6.830E-5;...
      2.540E-4, 2.160E-4, 1.210E-4, 5.600E-5 ; 2.640E-4, 2.170E-4, 1.160E-4,
4.960E-5;...
      2.740E-5, 2.200E-4, 1.160E-4, 4.550E-5];

tt = temp - 273.15;  % temp in celsius
np = 0;
npp = 0;

if (freq > 0)
    lambda = c * 1e-6 / freq;    %wavelength in mm

    % Interpolate to estimate complex index
    if ( lambda > wl(1) && lambda < wl(45) )
        j = 2;
        while( lambda > wl(j) &&  j < 45 )
            j = j+1; % Yields bounds between j and j-1
        end
        wwght = (log(lambda)-log(wl(j -1)))  / (log(wl(j))-log(wl(j - 1)));

        if ((tt <= 0) && (tt > -80))
            if (tt > temp_data(1))
                % Temp bounded below : extrapolate above
                twght = (tt-temp_data(2)) / (temp_data(1)-temp_data(2));
                np = wwght*(twght* re(j,1) + (1-twght) * re(j , 2)) + (1-
```

```
wwght)*(twght*re(j - 1, 1)+...
                    (1-twght) * re(j - 1, 2));
                npp = wwght*(twght* im(j ,1) + (1-twght)*im(j , 2)) + (1-
wwght)*(twght*im(j - 1, 1)+...
                    (1-twght)*im(j - 1,2));
            else
                if (tt < temp_data(4))
                    % Temp bounded above : extrapolate below
                    twght = (tt-temp_data(4))/(temp_data(3)-temp_data(4));
                    np =  wwght*(twght*re(j,3)+(1-twght)*re(j,4)) + (1-
wwght)*(twght*re(j - 1,3)+...
                        (1-twght)* re(j - 1,4));
                    npp = wwght*(twght*im(j , 3)+(1-twght)*im(j ,4)) + (1-
wwght)*(twght*im(j - 1,3)+...
                        (1-twght)*im(j - 1,4));
                else
                    % Temp bounded above and below
                    i = 2;
                    while ((tt < temp_data(i)) && (i<4))  %Yields bounds
between i and i-1}
                        i = i + 1;
                    end
                    twght = (tt-temp_data(i))/(temp_data(i-1)-temp_data(i));
                    np =  wwght*(twght*re(j,i-1) + (1-twght)*re(j,i)) + (1-
wwght)*(twght*re(j - 1,i-1) + (1-twght)*re(j - 1,i));
                    npp = wwght*(twght*im(j , i-1) + (1-twght)*im(j,i)) + (1-
wwght)*(twght*im(j - 1,i-1)+(1-twght)*im(j - 1,i));
                end
            end
        else
             % Out of range temperature requested
            disp(sprintf('WARNING : Temperature %f (deg C) out of range in
"h2o_ice_diel.m"',tt));
            np  = 1;
            npp = 0;
        end % if tt <= 0
 else
    %Out of range frequency requested}
    disp(sprintf('WARNING : Wavelength %f (mm) out of range in
"h2o_ice_diel.m"',lambda));
    np  = 1;
    npp = 0;
 end
  % Convert to relative dielectric constant}
  n = np - npp * sqrt(-1);
  kappa = n ^ 2;
end
```

```
%%% Program Name: d3lec
%%% Description:
%%% 1. This program is for computing the permittivity of water
%%% 2. It is written by Sandeep Kumar, CET.
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function [kappa] = d3lec(FREQ, TEMP, SL, I)
format long e;
T = TEMP - 273.2;
SIGMA = 12.5664e8;
EI = 5.27137 + T * ( 0.0216474 - 0.00131198 * T );
T = T - 25.0;
ES = 78.54 * (1.0 + T * ( -4.579e-3 + T * ( 1.19e-5 - 2.8e-8 * T ) ) );
ALPHA = 0.0609265 - 16.8129 / TEMP;
LS = 0.33836e-5 * exp( 2513.98 / TEMP);
if (I ~= 2)
   S = SL * 1000.0;
   %Normality as a function of salinity [Klein & Swift]
   N = S * (1.707e-2 + S * (1.205e-5 + 4.058e-9 * S));
   DEL = -T;
   SIG = 0.0;
   if (I == 1)
       %Conductivity of sea water [Klein & Swift]
       SIG = S * ( 0.182521 + S * ( -1.46192e-3    ...
            + S * ( 2.09324e-5 - 1.28205e-7 * S ) ) )    ...
            * exp( - DEL * ( 2.033e-2 + DEL * ( 1.266e-4 ...
            + 2.464e-6 * DEL ) - S * ( 1.849e-5          ...
            + DEL * ( -2.551e-7 + 2.551e-8 * DEL ) ) ) );
   elseif (I == 0)
       %Conductivity of NACL solution [Stogryn - Equations for calculating
the diel
       %ectric constant of saline water
       SIG = N * ( 10.394 + N * ( -2.3776 ...
            + N * ( 0.68258 + N * ( -0.13538 ...
            + N * 1.0086e-2 ) ) ) ) ...
            * ( 1.0 + DEL * ( -1.962e-2 + DEL ...
            * 8.08e-5 ) - DEL * N * ( 3.020e-5 ...
            + 3.922e-5 * DEL + N * ( 1.721e-5 - 6.584e-6 * DEL ) ) );
   end
   SIGMA = SIG / 8.854e-12;
   % ES = ES * a(N) - Stogryn
   ES = ES * ( 1.0 + N * ( -0.2551 + N * ( 5.151e-2 ...
         - 6.889e-3 * N ) ) );
   % LS = LS * b(N) - Stogryn
   LS = LS * ( 1.0 + N * ( T * 0.1463e-2 - 0.04896 ...
         + N * ( -0.02967 + N * 5.644e-3 ) ) );
end

S = sin(ALPHA * 90.0 * (pi/180));
L = 2.99776e-1 / FREQ;
T = (LS/L) ^ (1.0 - ALPHA);
D = 1.0 + T * (2.0*S + T);
EP = EI + (ES-EI) * (1.0+ T*S) / D;
EPP = (ES-EI) * T * cos(ALPHA*90.0 * (pi / 180))/D + L*SIGMA/18.8496e8;
eps = EP - EPP * sqrt(-1);
kappa = eps;
```

```
%%% Program Name: mie2
%%% Description:
%%% 1. This program is for computing Mie coefficients for given complex
%%%    permittivity and permeability ratios
%%% 2. It is written by C. Matzler, July 2002
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function result = mie2(eps1, mu1, x)
if x == 0                % To avoid a singularity at x=0
    result = [0 0 0 0 0 1.5];
elseif x>0              % This is the normal situation
    nmax = round(2+x+4*x.^(1/3));
    n1 = nmax-1;
    n = (1:nmax);cn=2*n+1; c1n=n.*(n+2)./(n+1); c2n=cn./n./(n+1);
    x2 = x.*x;
    f = mie2_ab(eps1,mu1,x);
    anp = (real(f(1,:))); anpp=(imag(f(1,:)));
    bnp = (real(f(2,:))); bnpp=(imag(f(2,:)));
    g1(1:4,nmax) = [0; 0; 0; 0]; % displaced numbers used for
    g1(1,1:n1) = anp(2:nmax);    % asymmetry parameter, p. 120
    g1(2,1:n1) = anpp(2:nmax);
    g1(3,1:n1) = bnp(2:nmax);
    g1(4,1:n1) = bnpp(2:nmax);
    dn = cn.*(anp+bnp);
    q = sum(dn);
    qext = 2*q/x2;
    en = cn.*(anp.*anp+anpp.*anpp+bnp.*bnp+bnpp.*bnpp);
    q = sum(en);
    qsca = 2*q/x2;
    qabs = qext-qsca;
    fn = (f(1,:)-f(2,:)).*cn;
    gn = (-1).^n;
    f(3,:) = fn.*gn;
    q = sum(f(3,:));
    qb = q*q'/x2;
    asy1 = c1n.*(anp.*g1(1,:)+anpp.*g1(2,:)+bnp.*g1(3,:)+bnpp.*g1(4,:));
    asy2 = c2n.*(anp.*bnp+anpp.*bnpp);
    asy = 4/x2*sum(asy1+asy2)/qsca;
    qratio = qb/qsca;
    result = [qext qsca qabs qb asy qratio];
end;
```

```
%%% Program Name: mie2_ab
%%% Description:
%%% 1. This program is for computing Mie coefficients for given complex
%%%    permittivity and permeability ratios
%%% 2. It is written by C. Matzler, July 2002
%%% The code is last modified by Miao Tian, CET, 07/06/2012.

function result = mie2_ab(eps1,mu1,x)

m = sqrt(eps1.*mu1);           % refractive index ratio
z = m.*x;
z1 = sqrt(mu1./eps1);          % impedance ratio
nmax = round(2+x+4*x.^(1/3));
nmx = round(max(nmax,abs(z))+16);
n = (1:nmax); nu = (n+0.5);
sx = sqrt(0.5*pi*x);
px = sx.*besselj(nu,x);
p1x = [sin(x), px(1:nmax-1)];
chx = -sx.*bessely(nu,x);
ch1x = [cos(x), chx(1:nmax-1)];
gsx = px-1i*chx; gs1x=p1x-1i*ch1x;
dnx(nmx) = 0+0i;
for j = nmx:-1:2      % Computation of Dn(z) according to (4.89) of B+H (1983)
    dnx(j-1) = j./z-1/(dnx(j)+j./z);
end;
dn = dnx(n);
da = dn.*z1+n./x;
db = dn./z1+n./x;

an = (da.*px-p1x)./(da.*gsx-gs1x);
bn = (db.*px-p1x)./(db.*gsx-gs1x);

result=[an; bn];
```